6-1-2006

Computation Reuse in Statics and Dynamics Problems for Assemblies of Rigid Bodies

Anne Loomis
Dartmouth College

Master's thesis: computation reuse in statics and dynamics problems for assemblies of rigid bodies

Anne Loomis
Advisor: Devin Balkcom

Dartmouth College Technical Report
TR2006-576

Abstract

The problem of determining the forces among contacting rigid bodies is fundamental to many areas of robotics, including manipulation planning, control, and dynamic simulation. For example, consider the question of how to unstack an assembly, or how to find stable regions of a rubble pile. In considering problems of this type over discrete or continuous time, we often encounter a sequence of problems with similar substructure. The primary contribution of our work is the observation that in many cases, common physical structure can be exploited to solve a sequence of related problems more efficiently than if each problem were considered in isolation.

We examine three general problems concerning rigid-body assemblies: dynamic simulation, assembly planning, and assembly stability given limited knowledge of the structure's geometry.

To approach the dynamic simulation and assembly planning applications, we have optimized a known method for solving the system dynamics. The accelerations of and forces among contacting rigid bodies may be computed by formulating the dynamics equations and contact constraints as a *complementarity problem* [34]. Dantzig's algorithm, when applicable, takes n or fewer *major cycles* to find a solution to the linear complementarity problem corresponding to an assembly with n contacts. We show that Dantzig's algorithm will find a solution in $n - k$ or fewer major cycles if the algorithm is initialized with a solution to the dynamics problem for a subassembly with k internal contacts.

Finally, we show that if we have limited knowledge of a structure's geometry, we can still learn about stable regions of its surface by physically pressing on it. We present an approach for finding stable regions of planar assemblies: sample presses on the surface to identify a *stable cone* in *wrench space*, partition the *space of applicable wrenches* into stable and unstable regions, and map these back to the surface of the structure.

Acknowledgments

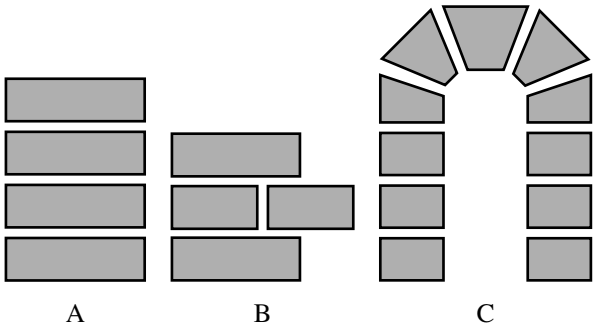
Special thanks to my advisor Devin Balkcom for all his input and support, as well as my committee, Amit Chakrabarti, Chris Bailey-Kellogg, and Peng Song, and the Dartmouth Robotics Laboratory, for their helpful feedback, and Amit Chakrabarti for his insight into the simplex method. In addition, many thanks to Paritosh Kavathekar, for his help solving some ordinary differential equations, to Matt Bell, for his help ordering equipment, and to Steven Gomez, for his help pushing on bricks. Finally, to Scout Sinclair, Chris Masone, and Joshua Brody, thanks for being the best brain a girl could have.

This research program is a part of the Institute for Security Technology Studies, supported by Grant No. 2005-DD-BX-1091 awarded by the Bureau of Justice Assistance. The Bureau of Justice Assistance is a component of the Office of Justice Programs, which also includes the Bureau of Justice Statistics, the National Institute of Justice, the Office of Juvenile Justice and Delinquency Prevention, and the Office for Victims of Crime. Points of view or opinions in this document are those of the author and do not represent the official position or policies of the United State Department of Justice.

Contents

List of Tables

List of Figures



(a) Three examples of stable piles

(b) Rubble pile at New Jersey Task Force One field exercise.

Figure 1: Simple and complex multibody systems.

1 Introduction

The problem of determining the forces among contacting rigid bodies is fundamental to many areas of robotics, including manipulation planning, control, and dynamic simulation. For example, consider the problem of unstacking any of the structures in figure 1(a), or finding stable regions of the rubble pile in figure 1(b). We might also ask how an assembly moves under certain external forces, for applications in mechanical engineering or computer graphics. In considering problems of this type over discrete or continuous time, we often encounter a sequence of problems with similar substructure. The primary contribution of this thesis is the observation that in many cases, common physical structure can be exploited to solve a sequence of related problems more efficiently than if each problem were considered in isolation.

We examine three general problems concerning rigid-body assemblies: dynamic simulation, assembly planning, and assembly stability given limited knowledge of the structure's geometry. Our approach to both dynamic simulation and assembly planning is based on a theoretical result regarding computation reuse in a known method for solving the system dynamics.

- **Dynamic simulation.** During simulation, the positions of several bodies might remain fixed throughout a sequence of frames; figure 2 shows an example. We present an algorithm that computes the dynamics to identify the set of motionless bodies, re-computes the dynamics of that substructure, and uses the result as a starting point to compute the dynamics of the complete structure in successive time steps.

- **Assembly planning.** In many manipulation planning problems, some parameters or conditions are held constant while other conditions are not. For example, consider the problem of finding a sequence in which to disassemble the simple five-block structure shown in the upper left corner of figure 3 so that it does not collapse. At one point in the planning algorithm it may be necessary to analyze the stability of the structure $\{1, 3, 5\}$; at another point, it may be necessary to analyze the stability of the structure $\{1, 3, 4, 5\}$. We show that it is possible to reuse the results of the stability computation for the smaller structure to compute the stability of the larger structure more quickly.

- **Stability of assemblies.** In some real-world problems, we do not have complete information about the geometry of an assembly. However, we can still find stable regions of its surface if we know that the geometry of the structure does not change over time. We show that by applying known forces to the surface of an assembly, we can partition the space of possible wrenches into stable and unstable regions, and map this information back to the surface to define the assembly's stability profile. In addition, we prove the existence of assemblies for which no set of applied wrenches will reveal information about stability along other areas of the surface.

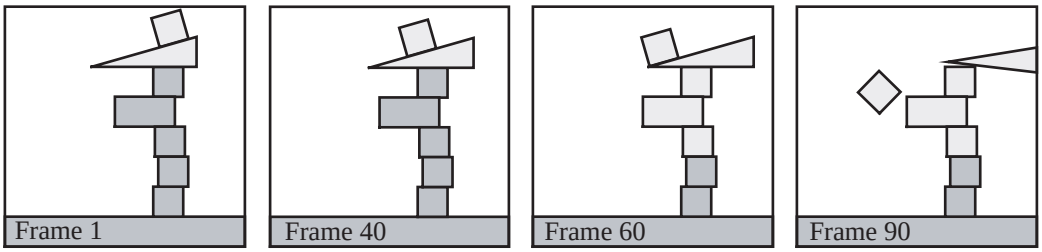


Figure 2: A column falling apart. Dark gray blocks are motionless. The motionless set changes at frame 60.

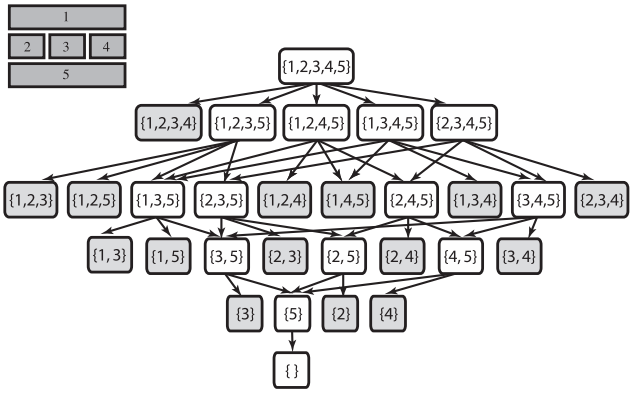


Figure 3: A simple structure and its disassembly graph.

The results we present in this thesis show that common physical substructure can be exploited to optimize computations across a series of related problems. We believe that this approach to rigid-body dynamics is broadly applicable to multibody systems.

In general, the level of complexity that arises when a complete assembly is considered can be prohibitive. For example, most robust methods for determining the stability of rigid-body assemblies have an exponential running time. The manipulation of multibody systems is something relatively few researchers have even considered. Of those who have [1, 10, 11], all have focused on developing a dynamics formulation and establishing the existence of a solution method; to our knowledge, they have not considered how common physical structure can be used to make solving these problems more efficient. It is our hope that the ideas presented in this thesis may provide tools for thinking about how to handle multibody problems more simply and with reduced levels of complexity. We hope this work takes steps toward making multibody grasping and manipulation a more manageable and accessible area of research.

1.1 Approach

Our approach to computation reuse is based on the observation that fundamental components of the system statics and dynamics have a certain structure. In solving system dynamics, we use this structure explicitly to optimize our solution methods; in the stability of assemblies problem, the structure is implicit, but our knowledge of it allows us to answer questions about stability that we would not have been able to address otherwise.

The typical model of contacting rigid bodies consists of the Newton-Euler dynamics equations, unilateral-force constraints, non-penetration constraints, and frictional constraints. With the correct choice of coordinates, the equations and constraints form a *complementarity problem*, which will be discussed in greater detail in section 3.3. One method for determining stability is to use the system dynamics to show that there exists a solution such that a structure is not in motion. In this thesis, we consider only *linear* complementarity problems (LCPs), such as those that arise in planar systems, in spatial systems without friction, and in spatial systems with linear approximations of Coulomb friction, such as those considered in [64, 67, 65].

Figure 4: Structures with identical surfaces can have very different stability profiles.

In a linear complementarity problem $(\mathbf{A}, \mathbf{b})$, we seek to find vectors $\mathbf{f}, \mathbf{a}$, that satisfy the system

$$\mathbf{a} = \mathbf{A}\mathbf{f} + \mathbf{b}, \tag{1}$$

$$\mathbf{a}, \mathbf{f} \geq 0, \tag{2}$$

$$\mathbf{a}^T\mathbf{f} = 0, \tag{3}$$

where $\mathbf{A}$ is a constant $n \times n$ matrix, and $\mathbf{b}$ is constant vector of length n. We call a solution to equation 1 that satisfies constraints 2 and 3 a *complementary feasible solution.*

The structure of the matrix $\mathbf{A}$ and the physical interpretation of the vectors depend on the particular dynamics formulation. For example, in [6], $\mathbf{a}$ is a vector of accelerations at the contact points, $\mathbf{f}$ is a vector of forces applied at the contacts, and

$$\mathbf{A} = \mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T \qquad \text{and} \qquad \mathbf{b} = \mathbf{J}\mathbf{M}^{-1}\mathbf{F}^{\text{ext}}, \tag{4}$$

where $\mathbf{M}$ is the mass matrix, $\mathbf{J}$ is the Jacobian relating motion of the bodies in generalized coordinates to motion of the contact points, and $\mathbf{F}^{\text{ext}}$ is the vector of external and velocity-dependent forces. In [64], $\mathbf{a}$ is a vector of velocities, and $\mathbf{f}$ contains impulses. In most formulations, $\mathbf{A}$ can be partitioned in such a way that each block corresponds to a particular contact present in the physical system.

There are a variety of solution methods for LCPs, some of which will be discussed in more detail in section 3. Of these methods, we apply the principle of computation reuse to Dantzig's *principle pivoting method*, which forms the basis for Baraff's approach to solving the system dynamics in [6]. In its unmodified form, Dantzig's algorithm performs a series of n or fewer *major cycles* to successively satisfy each constraint $a_i \geq 0, f_i \geq 0$ of equation 2. We show that the solution to the dynamics equations for a structure can be found in $n - k$ or fewer major cycles of Dantzig's algorithm if the solution for the dynamics equations are known for any substructure with k internal contacts. The size of k depends on the problem we consider: in the assembly planning application, k is typically equal to $n - 2$, so the algorithm runs in 2 or fewer major cycles, whereas in the dynamic simulation application, k may range from $n - 1$ to zero, depending on the structure we simulate. Although this result seems physically intuitive, it is not clear how computation can be similarly reused in other complementarity algorithms such as Lemke [32] or PATH [19]. We apply this computation reuse method to both the stability problem in section 4.3 and the dynamic simulation problem in section 4.4.

To calculate stability as described above, we must make the limiting assumption that we have perfect knowledge of the geometry of the system. This has a significant effect on our ability to analyze how to manipulate and interact with the physical world. In section 5, we consider how to approach computation reuse in the face of geometric unknowns.

Consider an assembly such as one of those shown in figure 4, and assume we know nothing about its geometry below the surface. How might we learn about the stability of the structure? One approach is to physically press on it. If it is stable under several presses, can we predict other areas of the surface where we can apply stable presses?

Our goal in section 5 is to take steps toward characterizing what we can learn about the stability of a structure by pressing on it. By a theorem given by Romney in [60], any linear combination of stable wrenches is stable. If we can find stable wrenches $\mathbf{W}_1$, $\mathbf{W}_2$, and $\mathbf{W}_3$ by experimentation, we can determine that a wrench $\mathbf{W}_4$ is stable if the linear

program

$$\mathbf{W}\mathbf{x} = \mathbf{W}_4, \mathbf{x} \geq \mathbf{0} \tag{5}$$

has a solution. Moreover, we can use the polyhedral convex cone defined by the stable wrenches to partition regions of wrench space into stable and unstable regions, which we can then map back to the surface of the assembly, to define its stabilit profile.

For a structure with a polygonal surface and two stable presses on it, we describe the magnitude of a press between them that we can guarantee to be stable. We give experimental evidence both that structures exist for which our magnitude function is precisely true, and for which it is conservative. For structures with edges that can be modeled with smooth parametric equations, we give the general form for the space of wrenches that can be applied to the surface, and discuss how to select presses that contain this space in a stability cone. We also present a theoretical result showing the existence of surfaces for which no information can be obtained about the stability of the surface by applying frictionless presses.

1.2 Thesis overview

In this section, we have presented the objectives of the thesis and our approach to accomplishing them. The primary focus of this research is to illustrate that, for many types of robotics problems, knowledge of forces among contacting rigid bodies can be used to quickly solve related problems with common substructure. In section 2, we describe work in the related areas of grasping, rigid body dynamics, simulation, and assembly planning, and highlight any methods for reusing computation the authors discuss. In section 3, we describe several formulations of rigid-body system dynamics, introduce some terminology regarding LCPs, and discuss two important methods for solving them. In section 4, we present our modification to Dantzig's algorithm, and describe how we have applied it to the problems of determining a *stable disassembly path* for a given structure (such as the ones shown in figures 5, 6(a) and 6(b)) and optimizing dynamic simulations, specifically those where the integration time step is small. In section 5, we give theoretical and experimental results for the stability problem with limited knowledge of geometry (figure 24, 4). At the end of sections 4 and 5, we discuss open problems and areas where these ideas may be applied further.

2 Related work

Sequences of systems involving common physical structures can be found in a wide range of computational problems. This section presents related work in the areas of stability, assembly planning, and dynamic simulation. Within stability we consider approaches to stabilizing single parts as well as assemblies, from the areas of grasping, fixturing, and manipulation. In the area of dynamic simulation, we discuss a number of ways researchers have formulated the system dynamics, along with various approaches to solving the different formulations. Within assembly planning we look at methods for generating assembly sequences for a structures, and ways of optimizing these methods.

In many of the papers we consider in this section, the authors are aware that common substructure is present in the problems they are solving, and may discuss ways of exploiting this to improve the efficiency of their methods. Yet with few exceptions (notably [73, 29]), computation reuse is considered only as an implementation detail, and mentioned more as an afterthought than as an observation that carries a deep fundamental significance. Throughout the following discussion we will highlight where researchers have noted the possibility of computation reuse, and in some cases, where they have not, but where methods are known for employing it.

2.1 Stability

Stability is a fundamental and well-studied problem in robotics and mechanical design, with applications in grasping and manipulation, assembly planning and fixturing. A number of researchers have given characterizations of stability; in general, we define a structure as stable if it does not collapse under gravity, or some set of external forces. The following discussion focuses first on how researchers have addressed stabilizing a single workpiece, and then considers stability of assemblies.

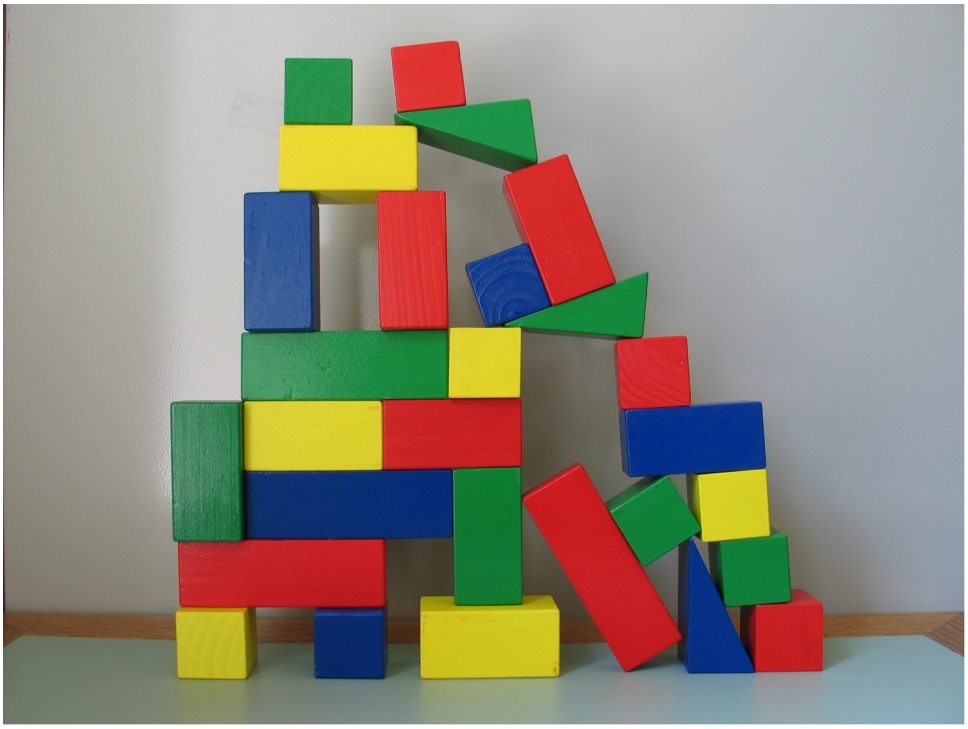


Figure 5: Stable pile of blocks with non-obvious unstacking order.

The robotics community generally recognizes the terms *form closure* and *force closure* to describe the immobility of an object held in a multi-fingered grasp. Rimon and Burdick discuss the relationship between these concepts and give precise definitions for them in [58]. Practically speaking, an object that is immobilized completely by the geometry of the grasp on it is held in form closure, whereas a force closure grasp stabilizes the object against any set of external forces. For a frictionless grasp, the concepts are equivalent; for friction, force closure grasps are a superset of form closure grasps – the friction in the system may provide force closure where form closure is not present. Early work by Mishra *et al.* gave sufficient conditions and algorithms for placing frictionless fingers to ensure a stable grasp [43, 44] with respect to a force or set of forces. Rimon and Burdick considered immobility under the less restrictive notion of second-order geometry, *i.e.*, where the surface normals indicate the object is not immobilized, but the curvature of the surface prevents any kind of motion [57]. Nguyen [50] developed means for constructing grasps by identifying regions on the perimeter of a part where, by placing fingers in those regions, the part would be held in form or force closure. Erdmann's [21] work on representing frictional contacts in configuration space is also seminal. Recent work by Cheong *et al.* considers the problem of grasping a collection of polygons connected by hinges [14].

One approach to determining stability is to solve the system dynamics: if a structure is in motion, it is unstable. However, as will be discussed in the following section, if multiple solutions exist to the dynamics equations, it is unclear whether to designate the structure as stable or unstable. Several researchers have proposed different characterizations for stability to address this problem, known as indeterminacy. Palmer [52] analyzed the computational complexity of calculating stability for planar rigid polygons. He defined two simple classifications: *potential stability*, in which a solution for the contact forces in an assembly satisfies the equilibrium equations and friction laws, and *guaranteed stability*, in which such a solution exists when friction is neglected. Both of these problems are in P, and can be easily tested by solving a linear program. His third classification, *infinitesimal stability*, in which there is no infinitesimal motion for which the configuration is unstable, he showed to be NP-hard.

More recently, the problem of finding sufficient conditions for stability with friction has been explored by Pang and Trinkle [54, 68], who revise Palmer's notions of potential and guaranteed stability, calling them overly restrictive. In their place, the authors define *weak* and *strong* stability. A system of rigid bodies is *weakly stable* if there exist

feasible contact forces such that there exists a solution to the equilibrium equations consistent with immobility. A system is *strongly stable* if there do not exist feasible contact forces consistent with motion. In this system, weak stability corresponds to Palmer's potential stability, while the set of strongly stable loads strictly contains the Palmer's set of guaranteed stable loads. Balkcom and Trinkle [4] provide a complete, albeit exponential, solution for computing a set of external wrenches with regard to which a single planar block is strongly stable.

Given the complexity of computing strong stability, it is common in assembly planning to focus exclusively on finding weak stability for an assembly. We adopt this convention in the following discussion.

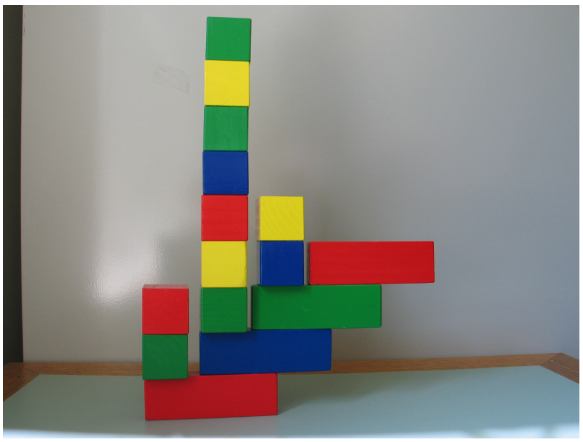
Blum *et al.* developed and implemented a stability test based on the well-known static equilibrium equations in [12]. As discussed in previous sections, this approach uses the geometry of an assembly to determine whether contact forces between objects can balance the known external gravitational force. If a solution exists, the structure is said to be stable. This problem can be formulated as a linear program (LP), and solved by testing the feasibility of the LP, which has the same asymptotic complexity as finding a solution to the linear program [17]. There are a variety of solution methods for linear programs, including the simplex method, which in the worst case can run in exponential time, but tends to be quite efficient in practice. Interior-point methods such as the one discovered by Karmarkar [30] can be proven to run in polynomial time, but seldom improve on the performance of simplex in practice. A variety of methods are known for reusing computation in simplex implementations. Starting simplex with an optimal basis from a similar problem may reduce the number of iterations it takes to find an optimal solution [24]. Within the same problem, there are ways to use an old basis to solve systems of equations involving the current basis [18] or to update LU-factorizations between pivots [23], which are described by Vanderbei in [70].

Boneschanscher *et al.* [13] explore the problem of subassembly stability, and give an algorithm for computing potential stability of stable orientations of an assembly on a table, by means of a series of culling tests. The first test assumes all parts in the assembly are attached and prune unstable orientations by determining whether the center of mass is over the convex hull of the assembly's support polygon. The next considers parts as unattached and converts the geometric model into a network. Then, for each part, the network is reduced to a relationship between the part and the table, and a linear program is solved to determine potential stability of the part. This is an improvement over prior methods such as the one presented by Blum *et al.* [12], since it can indicate which part in an assembly is unstable and can address external insertion forces in addition to gravitational forces. However, if there are loops in the contact graph, which is a characteristic of many assemblies, the geometry of the assembly must be altered for the algorithm to work correctly.

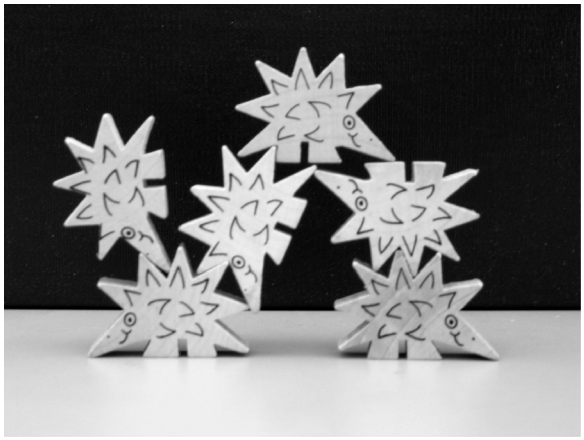
Mattikalli *et al.* consider the problem of finding potentially stable orientations of assemblies. Such work is useful for assembly planning, because reorienting a structure during its assembly can reduce the need for complicated and costly fixtures to maintain stability, and may reduce sensing costs for determining where objects are after manipulation. In [38], the authors present a method for finding a stable orientation of a frictionless assembly, or the "least unstable" orientation if no stable one exists. They formulate the problem a constrained maximin problem, and show that the most stable orientation can be found by solving a single linear program. In [39], the authors build on this work to give an exact characterization of the entire set of stable orientations for an assembly with Coulomb friction under gravity. They use parametric programming and sensitivity analysis techniques to determine the vertices and traverse the perimeter of the stability region.

Mosemann *et al.* [48] address several shortcomings of Mattikalli's algorithm and give an alternate approach for finding the set of all potentially stable orientations of an assembly with friction. They argue that Mattikali's method is not guaranteed to terminate and does not produce the magnitudes of forces required to make an assembly stable. They formulate the three-dimensional stability problem as a linear program and enumerate all vertices of the space of feasible solutions. The complexity of their algorithm is $O((m+n)mn\binom{n-2}{m-1})$, which is exponential in m, the number of objects, and n, the number of contacts, in the assembly.

Bernheisel and Lynch are among few researchers who have explored the manipulation of multi-object assemblies. In [10], the authors describe a graphical method that gives conservative bounds on pushing motions which guarantee the stability of a planar stack of objects during the push. Their approach finds the set of all stable motions for a stack by intersecting the set of all stable pushes on each part with the sets of stable pushes for objects below it in the stack. This method solves a series of subproblems where each part in the stack is pushing the parts ahead of it. In [11], the authors propose a linear constraint satisfaction problem formulation for determining stable pushes of a planar stack. If no solution exists, the assembly is guaranteed to be unstable for the given pushing motion. If any force necessary



(a) Object masses can affect stability.



(b) Object geometries can affect stability.

Figure 6: Several interesting assemblies.

to assure stability of the system can be generated as a linear combination of the pushing contacts, then it is possible to generate weakly stable pushes of the system. The method to determine this requires considering all 8^n combinations of support wrenches. If a stable push is neither impossible nor always possible, then the stack is classified as undecided.

A survey of stability in the context of fixturing can be found in [27]; with regard to grasping and manipulation, Mason gives an excellent discussion of geometric and algebraic methods for finding stable configurations for objects in [36].

Various other methods for computing stability are employed that are specific to their application. Ayyadevara *et al.* have developed a stacking planner and methods for generating interference-free part configurations by minimizing a cost function [3]. For robotic excavation, Singh *et al.* [61] describe a voxel model of soil to analyze stability during planning and execution of digs. For self-reconfiguring robots, Kotay *et al.* have precomputed motion sequences to ensure dynamic stability at each step [31]. Unsal *et al.* propose verifying that the center of mass of a self-reconfiguring robot held together by link joints remains over its convex hull as part of a decision-making algorithm for the robot's motions [69].

Certain special cases of assemblies have been completely analyzed, including the popular unstacking game of Jenga, for which Zwick [75] gives simple sufficient conditions for stability in the case where there are three blocks in a row. Paterson *et al.* [55] use a linear programming approach to determining stability to solve the problem of overhang, or how far off the edge of a table a stack of blocks can reach.

In this thesis, we are concerned with determining stability across a sequence of related problems. We show that by exploiting a knowledge of common substructures, we can improve on the linear programming approach to stability. Additionally, to our knowledge, we are the first researchers to propose a method for determining stability against external forces when the geometry of the structure beneath the surface is unknown.

2.2 Rigid-body dynamics

Computing the forces between contacting bodies is a fundamental problem in rigid-body dynamics, and integral to any understanding of how a robot interacts with its environment, whether the goal is to solve optimal control problems, plan robot manipulation tasks, or determine the stability of an assembly [67]. Because rigid-body models require that both forces and accelerations at contact points be non-negative, *i.e.* prevent penetration, the constraints forces that arise are unilateral, which result in systems of linear or nonlinear inequalities. This turns out to be a harder problem than computing bilateral constraint forces, *e.g.* forces imposed by joints, for which there are linear-time algorithms for loop-free systems [7, 22].

Lötstedt was the first to show that if the system constraints are inequality constraints the system dynamics can be modeled as a quadratic program [35] or a linear complementarity problem [34]. Typically, these problems are formulated in such a way that their solution represents accelerations and contact forces between bodies in the system. For the frictionless case, such problems always have a solution, and if $\mathbf{A}$ is nonsingular, then that solution is unique.

For most applications of practical interest, a frictionless model is not sufficient, but introducing friction into the model adds a host of complications. As first observed in 1895 by Painlevé [51], the dynamics equations for a Coulomb friction model may have multiple solutions (indeterminacy) or no solutions at all (inconsistency). One-contact point configurations that demonstrate indeterminacy and inconsistency are explored in [5, 21, 34, 37].

For much of the early 1990's, researchers formulated and solved constraint equations in order to cope with these problems in simulating the frictional model. Baraff showed in [5] that the problem of computing contact forces for consistent configurations is NP-hard if applying impulsive forces to consistent configurations is prohibited. By relaxing this assumption and allowing for impulsive forces in consistent configurations, he was able to interpret a result from Lemke's algorithm that does not find a solution (*i.e.* termination in an unbounded ray) as an impulsive force, thus finding a "valid contact impulse" according to his definition of the term, in what is considered polynomial time in practice. However, using Lemke's algorithm, static and dynamic friction must be handled as separate cases. Baraff is able to unify his treatment of static and dynamic friction by using Dantzig's principle pivoting method for solving the dynamics LCP, with modifications to maintain auxiliary friction constraints [6].

Trinkle *et al.* propose a model that approximates the nonlinear friction cone from Coulomb friction with a friction pyramid in [67]. Their resulting LCP always has a solution for small enough values of the friction coefficient μ. For arbitrary values of μ or rank-deficient system Jacobians, however, their solution methods are not guaranteed to converge. Other researchers work around limitations in the rigid-body model by allowing bodies to deform slightly. Witkin and Welch eliminate inconsistency and indeterminacy in configurations with one contact point by allowing linear and quadratic deformations of a shape, but the problems remain for multiple contact points [74].

In contrast to the traditional acceleration/contact force approach, more recent formulations try to sidestep the problem of inconsistency by formulating the system dynamics so that their solutions represent velocities and contact impulses. By considering the integrals of force *over a time step*, these methods allow an impulse to occur at any point in the duration of contact, rather than at the precise moment of impact. Stewart and Trinkle build on the work of Moreau [46], [47] and Monteiro-Marques [45] to develop an implicit time-stepping scheme in which a mildly nonlinear complementarity problem is solved using a sequence of LCPs [64], which are guaranteed to have a solution when contact normals are linearly independent. In [1], Anitescu and Potra extend this result so that the formulation is guaranteed to have a solution regardless of configuration or number of contacts. These models may achieve the desired degree of accuracy by adding edges to the linear approximation of their friction cones, however the complexity of the algorithm is exponential in the number of edges. Pang and Stewart discuss techniques that use general convex friction cones, but their formulation is highly nonlinear, making it difficult to find solutions for [53]. Anitescu *et al.* give an overview of time-stepping methods in [2], and Stewart summarizes much of the current research on rigid-body dynamics with friction and impact in [65].

2.3 Solution methods for simulation

All of the above formulations are complementarity problems, for which there are a variety of solution methods. More of them than can be covered here are discussed in detail in Cottle *et al.* [16] and Murty [49], both of which provide excellent coverage of LCPs in general. Here we consider two different types of approaches to solving an LCP: those where physical accuracy is desired, which are well-suited to applications such as fixturing and mechanical design where accuracy is valued over speed; and those used in graphics and animation, when a fast algorithm that produces a visually believable simulation is sufficient.

Within the robotics and mechanical simulation communities, attention has been primarily focused on Lemke's complementary pivot algorithm [32] and Dantzig's principle pivoting method [15] for LCPs, about which solution properties may be proven for certain classes of matrices. Dantzig's algorithm is guaranteed to find a solution if one exists, and $\mathbf{A}$ is either a P-matrix (has positive principle minors) or is positive semi-definite. Lemke's algorithm is applicable to a somewhat larger class of problems, and guaranteed to find a solution if $\mathbf{A}$ is a P-matrix. If $\mathbf{A}$ is copositive-plus, termination of Lemke's algorithm in an unbounded ray indicates inconsistency of the system, which

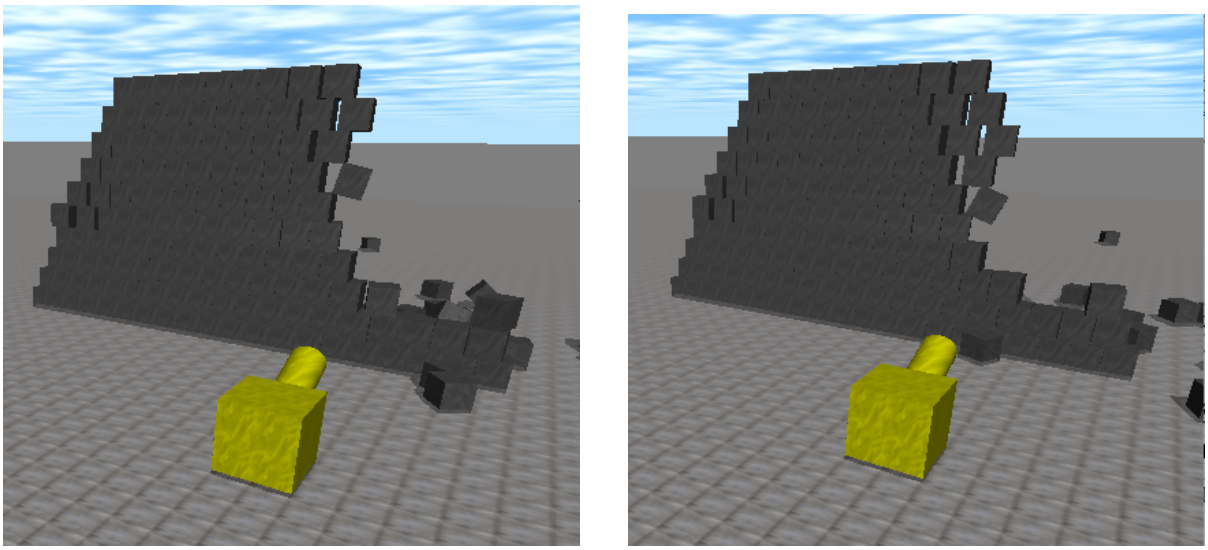


Figure 7: Aphysical configurations can result from the disabling function in ODE: these images show an assembly in its final resting state, when simulated using ODE's StepFast method. Used with permission of Russell Smith.

implies that Lemke's algorithm will find a solution if one exists, or indicate that no solution exists. Cottle, Pang, and Stone [16] make additional guarantees for copositive matrices, specifically that if $\mathbf{A}$ is copositive and $\mathbf{b}$ is in the dual cone of the homogeneous LCP, a solution must exist and Lemke's algorithm will find it. Stewart and Trinkle use this property to prove the existence of solutions for their formulation [64]. In practice, Dantzig and Lemke often converge even in the case where $\mathbf{A}$ is not of a suitable class. For nonlinear complementarity problems (NCPs), researchers often use Dirkse's PATH solver [19], which is generally regarded as a robust solver, although instances are known in which the PATH solver has not converged [66].

In [5], Baraff discusses a case where computing the solution to an LCP may be sidestepped entirely. He observes that the index set denoting the basic solution to the assembly's LCP usually remains valid across a number of time steps. As long as this is the case, the contact force vector may be found simply by solving a linear equation. This method of computation reuse is different from the approach we employ in section 4.4, when we reuse the solution to an LCP for a motionless substructure to more quickly compute the dynamics of the complete assembly.

When the physical accuracy of a simulation is not of primary importance, much faster solution methods may be used. It is instructive to look at solution methods in industry-standard physically-based modeling systems, for example, Russell Smith's Open Dynamics Engine (ODE) [62], which is generally considered to be the best open-source physics engine available. ODE implements Baraff's fast contact force computation algorithm [6], which, in the frictionless case, reduces to Dantzig's algorithm; with friction, the problem is not strictly an LCP, and the algorithm is not guaranteed to converge, although it is thought to be robust in practice. As the problem size grows beyond a dimension of a few hundred (for three-dimensional problems, this may happen when as few as twenty blocks are in contact), the solution time may become intractable, the matrix size too large, or round-off errors may cause the algorithm to proceed incorrectly, a problem that affects all pivoting methods [16]. Because of this, ODE provides several optimization methods that trade accuracy for speed and substantially less memory use. The first is StepFast, which makes the assumption that the effect of every contact is localized, and uses Baraff's algorithm to solve the LCPs for much smaller problems ([62], Chapter 8). The experimental automatic disabling functionality associated with this approach, however, can lead to unconvincing configurations as shown in figures 7(a) and 7(b). These images are screenshots taken of an assembly at rest, when the system dynamics were computed using the StepFast method. A faster approximation method is given by ODE's QuickStep method, which uses a successive overrelaxation (SOR) method to solve the LCP, which introduces some degree of error into its solutions. In general, the class of iterative

methods, which include SOR's and interior-point methods, is considered better for solving large-scale LCPs, as are less sensitive to round-off error, and better problem structure such as sparsity. However, they typically converge only in the limit [16].

Other methods used in computer graphics to find contact force magnitudes between rigid bodies quickly include using a time step equal to the frame rate of the simulation [42], and propagation methods for contact resolution. Guendelman *et al.* propose a method which consists of determining contacts by sequentially allowing individual objects to fall for a small time increment, holding the remaining objects stationary. The resulting contact graph expresses contact relationships and dependencies between bodies. They then iterate through the contacts in an order suggested by the graph, and solve for the force magnitudes over several passes. In order to produce a plausible simulation in a small number of iterations, they complete their contact force calculation with a shock-propagation step, which forces stationary objects to have a velocity of zero. This algorithm is capable of solving for contact forces for up to a thousand objects in a matter of minutes per frame [26], but it is not physically accurate.

Given the speed of the algorithms used in simulations whose applications are primarily in computer graphics [62, 42, 26], it is important to remark that the goal of this thesis is not to improve upon these results (it has not been explored whether our techniques would provide an effective means of doing so). Our focus is to provide a means for improving the efficiency of algorithms that provide exact solutions. We consider our results to be most useful for simulating small-to-medium sized systems, where the relative position between a subset of objects remains fixed across a number of time steps. This is most likely to be true for simulations where the integration time step is small.

2.4 Assembly planning

Subassembly stability strongly affects the order in which an assembly may be put together or taken apart. For most assembly tasks, it is a requirement that the assembly remain stable after any part or set of parts is added. Many researchers in the area of automated assembly planning have considered generating assembly sequences.

In [40], de Mello *et al.* show that an AND/OR graph can be used to represent all possible assembly plans for a given structure. In [41], the authors give an algorithm for generating such a graph, which runs in polynomial time for weakly connected assemblies and exponential time for strongly-connected assemblies.

As structures grow in the number of components, the search space for an assembly sequence becomes intractable very quickly. In an industrial setting, it is rare that all assembly sequences are considered before selecting one that will be used, however there is concern among researchers that methods used to prune the search space be chosen carefully, lest they cull a good sequence prior to optimization. Heemskerk [28] has suggested heuristics that generate sequence plans by collecting parts that can be assembled as one group. Primarily, however, such methods are concerned with kinematic constraints such as mating characteristics of parts [20], or how different parts block each others' insertion. Wilson *et al.* optimize the disassembly sequence algorithm by returning a *precedence expression* when their planner makes queries the regarding moveability of parts. Precedence expressions are boolean rules that are evaluated to determine whether a given part is moveable. Evaluating the expression is typically faster than performing a geometric check on the remaining assembly to determine moveablility. Hoffman [29], too, considers the freedom of motion of subassemblies, and shows how freedom calculations can be reused for other spatial configurations of curved components. While Wilson and Hoffman recognize the importance of common substructure geometry in making disassembly sequence planners more efficient, neither of them consider computation reuse for determining the stability of a structure.

In [72], Wilson and Latombe use the moveability constraint to produce a polynomial-time algorithm for monotone, binary assembly sequences using straight-line insertions, by employing the *non-directional blocking graph* data structure. In [71], the authors address the more general *partitioning problem*, for which the goal is to find a proper subset of objects in an assembly that can be removed without violating a non-penetration constraint, and show that in its full generality, the problem is NP-complete. A structure that can be partitioning in this way is sometimes said to be able to be taken apart with two hands. Snoeyink has discovered a class of assemblies that cannot be taken apart with two hands [63].

In contrast to earlier researchers, Romney considers the issue of stability alongside that of sequencing. In [59], he presents a method for concurrently generating an assembly sequence and an associated fixture to hold subassemblies in place during assembly. He is concerned primarily with *insertion-force stability (IFS)*, or the ability of a subassembly

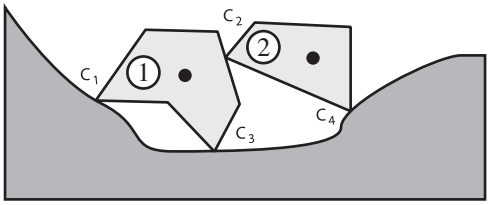


Figure 8: Two polygonal objects supported by the ground.

to remain stable when a part is inserted. He observes that IFS-checks may be cached for reuse, and gives conditions on the circumstances under which different subassemblies may reuse the same stability result [60]. Romney's results will be discussed in greater detail in section 5.

3 Physical model and solution methods

In the following section, we discuss how to formulate statics and dynamics equations from the geometry of the system. The statics formulation is a linear program (LP), and the dynamics equations, as given in [6], form a linear complementarity problem (LCP). Because we are concerned primarily with computation reuse in the dynamics formulation (which we use in section 4 both to determine stability and simulate the motions of the system), we look in depth at LCPs and several methods for solving them. First, we give an overview of the LCP, some common terminology regarding its solutions, and a geometric interpretation of the problem. We consider a brute-force algorithm for solving an LCP, which runs in exponential time and is therefore impractical for solving the general problem. We then discuss both Lemke's complementary pivot method and Dantzig's principle pivoting method, focusing in greater detail on the latter. We conclude with an examination of the advantages and disadvantages of these and several other methods commonly used to solve LCPs.

3.1 Rigid-body model

In this section, we discuss how to formulate the system Jacobian $\mathbf{J}$ from the force-balance equations, and then show how it is used in the system statics equations. We then present Baraff's system dynamics model from [6].

3.1.1 System statics

We focus first on formulating the system statics for planar systems.

Consider the equations for static equilibrium of object 1 in figure 8. Let $\mathbf{n}_j$ be the unit contact normal representing the direction of the contact force at point j, and f_j be the magnitude of the force. Let $\mathbf{g}$ be the unit vector representing the orientation of gravity. Let $\tau_{\mathbf{n}_j}$ be the torque due to contact j. Summing the forces and torques,

$$f_1\mathbf{n}_1 + f_2\mathbf{n}_2 + f_3\mathbf{n}_3 + f_G\mathbf{g} = \mathbf{0} \quad (6)$$

$$\tau_{\mathbf{n}_1} + \tau_{\mathbf{n}_2} + \tau_{\mathbf{n}_3} = 0 \quad (7)$$

must be satisfied for object 1 to be at rest. We can collect the equilibrium equations for each body in the system into the matrix equation 8. If $f_j < 0$, the contact force is attractive. Since the rigid-body model adopts a non-penetration constraint, we require that $f_j \geq 0$. Thus we can formulate an LP from the static equilibrium equations

$$\mathbf{J}^T\mathbf{f} = \mathbf{F}^{\text{ext}}, \mathbf{f} \geq \mathbf{0} \quad (8)$$

where $\mathbf{J}^T$ is the transpose of the system Jacobian and describes the constraints imposed on each body by the contact forces, and $\mathbf{F}^{\text{ext}}$ is a vector of external forces acting on the system. For a frictionless system, $\mathbf{J}^T$ contains blocks of

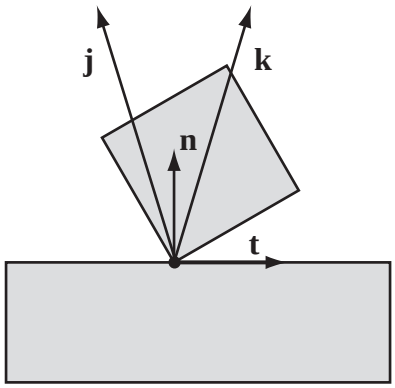


Figure 9: A contact point between two blocks. Contact normal, tangent, and left and right edges of the friction cone are shown.

the form

$$\mathbf{J}_{ij}^T = \begin{bmatrix} \mathbf{n}_{j_x} \\ \mathbf{n}_{j_y} \\ \mathbf{r}_{ij} \times \mathbf{n}_j \end{bmatrix} \tag{9}$$

where $\mathbf{r}_{ij}$ is the position vector from the center of mass of object i to contact point j, and $\mathbf{n}_j$ is the normal to the contact. To model Coulomb friction, the blocks of $\mathbf{J}^T$ take on the form

$$\mathbf{J}_{ij}^T = \begin{bmatrix} \mathbf{j}_{j_x} & \mathbf{k}_{j_x} \\ \mathbf{j}_{j_y} & \mathbf{k}_{j_y} \\ \mathbf{r}_{ij} \times \mathbf{j}_j & \mathbf{r}_{ij} \times \mathbf{k}_j \end{bmatrix} \tag{10}$$

where $\mathbf{j}_j$ and $\mathbf{k}_j$ are the left and right edges of the friction cone at contact point j, as shown in figure 9. Solving this equation corresponds to finding, for each contact, a force that is a positive linear combination of the left and right edges of the friction cone, *i.e.* lies within the friction cone. Thus, if a solution exists to the LP in system 8, it means that, at each contact point there is some force that satisfies the model of Coulomb friction and the entire system is at equilibrium.

An alternative formulation of $\mathbf{J}^T$ for the frictional model has blocks of the form

$$\mathbf{J}_{ij}^T = \begin{bmatrix} \mathbf{n}_{j_x} & \mathbf{t}_{j_x} \\ \mathbf{n}_{j_y} & \mathbf{t}_{j_y} \\ \mathbf{r}_{ij} \times \mathbf{n}_j & \mathbf{r}_{ij} \times \mathbf{t}_j \end{bmatrix} \tag{11}$$

where $\mathbf{n}_j$ and $\mathbf{t}_j$ are the normal and tangent vectors at contact point j. Formulating $\mathbf{J}$ in this way, system 8 is no longer applicable to determine the stability of the structure, as tangent force magnitudes $f_{\mathbf{t}}$ may be negative, and must satisfy the additional Coulomb friction condition $|f_{\mathbf{t}}| \leq \mu f_{\mathbf{n}}$ in order to lie within the friction cone. This is, however, the formulation that Baraff uses in [6], and thus represents the structure of $\mathbf{J}$ in the system dynamics model in section 3.1.2.

For planar structures, each row of $\mathbf{J}^T$ indicates how each one of the three degrees of freedom in two-dimensional space is constrained by the contact forces on a given body. We can extend this formulation to three dimensions by constraining each of the six degrees of freedom in three-dimensional space. Friction is handled by approximating the nonlinear friction cone from the Coulomb model of friction with a linearized friction cone, discussed in more detail in [65]. Depending on the application, this is a good approximation, since a small number of edges gives a reasonable degree of accuracy, and edges can be added to increase accuracy to the desired amount.

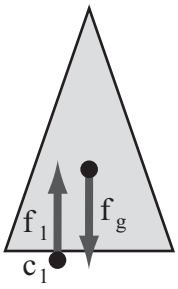


Figure 10: Example of a structure for which a necessary condition for stability, $\mathbf{Af} + \mathbf{b} = \mathbf{0}$, is satisfied, but the structure is not stable.

3.1.2 System dynamics

As in the prior section, let $\mathbf{n}_i$ be the contact normal between two bodies, and let f_i be the magnitude of the contact force between them. Recall that the rigid body model adopts a non-penetration constraint, so we require that $f_i \geq 0$. Similarly, let a_i be the relative acceleration between the bodies in the direction of $\mathbf{n}_i$, and require that $a_i \geq 0$. If the bodies are separating at a contact point, then there is no force between them. Conversely, if $f_i > 0$ the bodies remain in contact, so their relative acceleration is zero. Thus, a third constraint is that one of a_i or f_i must always be zero, or $a_i f_i = 0$. We collect a_i and f_i into the vectors $\mathbf{a}$ and $\mathbf{f}$, respectively.

The Newton-Euler equations describe the dynamics of the system, and give the linear relationship between $\mathbf{f}$ and $\mathbf{a}$ given in equation 1 where

$$\mathbf{A} = \mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T \qquad \text{and} \qquad \mathbf{b} = \mathbf{J}\mathbf{M}^{-1}\mathbf{F}^{\text{ext}}. \tag{12}$$

$\mathbf{J}$ is the system Jacobian, $\mathbf{M}$ is a block-diagonal matrix that describes the mass characteristics of the system, and $\mathbf{F}^{\text{ext}}$ is a vector that represents the net external force acting on the system, including all inertial velocity-dependent forces.

Equation 1, along with conditions

$$a_i \geq 0, f_i \geq 0 \qquad \text{and} \qquad f_i a_i = 0, \tag{13}$$

form an LCP (equation 13 is equivalent to equations 2 and 3). We can use this formulation to solve both the stability problem and to simulate the system dynamics.

To determine the stability of an assembly we need only solve the LCP. If a solution $(\mathbf{a}, \mathbf{f})$ is found that satisfies equations 2 and 3 such that 1) $\mathbf{a} = \mathbf{0}$ and 2) $\mathbf{f}$ satisfies the system's static equilibrium equations, $\mathbf{J}^T\mathbf{f} = \mathbf{F}^{\text{ext}}$, then the structure is weakly stable. (Figure 10 shows a structure for which the first condition, $\mathbf{a} = \mathbf{0}$, is satisfied but the equilibrium equations are not.) If a solution is found such that some $a_i > 0$, then we know that the structure is not strongly stable.

We can simulate the system dynamics by solving the LCP for the forces and accelerations at each time step. This will be discussed in greater detail in section 4.4.

3.2 Static friction

We add static friction to the model by augmenting the normal force conditions with several *static friction conditions*. The force due to friction at a contact point always acts in a direction tangent to the contact surface. Let a_{F_i} be the acceleration due to friction in the tangent plane at the ith contact, and let f_{F_i} be the magnitude of the friction force. At this point, we relabel a_i and f_i as a_{N_i} and f_{N_i}, to signify that they are in the direction of the contact normal, in contrast to a_{F_i} and f_{F_i}. Having introduced this notation, we can state the static friction conditions as follows. If there is no frictional acceleration at some contact i, then f_{F_i} must lie within the friction cone of the ith contact. Otherwise, in order to maintain the condition of static friction, *i.e.* that the structure has zero velocity at every contact point, we require that f_{F_i} does the maximum possible work to counter the frictional acceleration. Finally, we require that the friction force and acceleration are in opposite directions, that is, f_{F_i} and a_{F_i} must have opposite signs.

These conditions are summarized mathematically as

$$|f_{F_i}| \leq \mu f_{N_i}, \qquad a_{F_i}(\mu f_{N_i} - |f_{F_i}|) = 0, \qquad \text{and} \qquad a_{F_i} f_{F_i} \leq 0. \tag{14}$$

Baraff describes the necessary changes to Dantzig's algorithm to maintain these additional conditions in [6]. For our purposes, it suffices to say that the algorithm is essentially the same, except for the shape of the variables in the equation $\mathbf{Af} + \mathbf{b} = \mathbf{a}$. Each of $\mathbf{A}, \mathbf{f}, \mathbf{b}$ and $\mathbf{a}$ now contains frictional variables for every contact. We can arrange rows and columns so that the problem has the form

$$\mathbf{A} \begin{bmatrix} \mathbf{f}_{N_1} \\ \vdots \\ \mathbf{f}_{N_n} \\ \mathbf{f}_{F_1} \\ \vdots \\ \mathbf{f}_{F_n} \end{bmatrix} + \mathbf{b} = \begin{bmatrix} \mathbf{a}_{N_1} \\ \vdots \\ \mathbf{a}_{N_n} \\ \mathbf{a}_{F_1} \\ \vdots \\ \mathbf{a}_{F_n} \end{bmatrix}. \tag{15}$$

The necessary condition for stability is no longer $\mathbf{a} = \mathbf{0}$. Let $\mathbf{a}_N$ be the upper partition of $\mathbf{a}$ containing a_{N_i} for all i, and let $\mathbf{a}_F$ be the lower partition. Then for the assembly to be stable, we must have $\mathbf{a}_N = \mathbf{0}$ and $\mathbf{a}_F$ such that for all $a_{F_i} \neq 0$, $f_{F_i} = \mu f_{N_i}$.

3.3 Introduction to the LCP

In this section, we introduce some terminology necessary to our discussion of solution methods for LCPs. The organization of this section is loosely based on Chapter 1 of Murty [49], "Linear complementarity, its geometry and applications," with certain fundamental concepts augmented by Dantzig's introduction to linear systems in [17].

A linear complementarity problem consists of a matrix equation

$$\mathbf{a} = \mathbf{Af} + \mathbf{b}, \tag{16}$$

where $\mathbf{A}$ is a constant $n \times n$ matrix, $\mathbf{b}$ is constant vector of length n, and $\mathbf{a}$ and $\mathbf{f}$ are unknown vectors of size n. A *solution* is any value for $(\mathbf{a}, \mathbf{f})$ that satisfies equality 16. We seek to find a solution that satisfies the constraints

$$\mathbf{a}, \mathbf{f} \geq \mathbf{0}, \tag{17}$$

$$\mathbf{a}^T \mathbf{f} = 0. \tag{18}$$

We say that a solution $(\mathbf{a}, \mathbf{f})$ to equality 16 is *feasible* if it satisfies equation 17. One that satisfies equation 18 is called *complementary*.

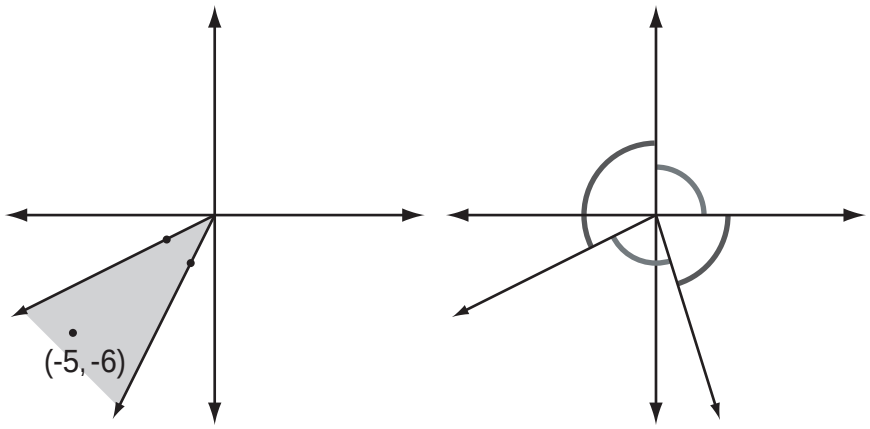
Most algorithms for solving LCPs partition the variables in $\mathbf{a}$ and $\mathbf{f}$ into two sets called basic and nonbasic variables. We describe these sets as follows. Consider the term $\mathbf{Px}$, where $\mathbf{P}$ is an $m \times n$ matrix. Without loss of generality, assume Rank($\mathbf{P}$) = m. We can write $\mathbf{Px}$ as

$$\mathbf{Px} = \mathbf{Bx}_B + \mathbf{Nx}_N \tag{19}$$

where $\mathbf{B}$ is a square, invertible, $m \times m$ submatrix of $\mathbf{P}$, and $\mathbf{N}$ is a matrix composed of the remaining columns of $\mathbf{P}$. The variables contained in $\mathbf{x}_B$ are called *basic* variables, and those contained in $\mathbf{x}_N$ are *nonbasic* variables. In a *canonical system*, the i^{th} basic variable has a unit coefficient in the i^{th} equation, and zero coefficients elsewhere. Equation 16 can be rewritten as the canonical system

$$\mathbf{Ia} - \mathbf{Af} = \mathbf{b}. \tag{20}$$

We call a solution to equation 16 a *basic* solution if all nonbasic variables have a value of zero. Every solution technique for an LCP seeks a *complementary basic feasible solution* for equation 16.



(a) A complementary cone for the LCP given by equation 22.

(b) The complete set of complementary cones of the matrix given by equation 25.

Figure 11: Two views of complementary cones.

Consider an example from Murty [49] where

$$\mathbf{A} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}, \qquad \mathbf{b} = \begin{bmatrix} -5 \\ -6 \end{bmatrix}. \tag{21}$$

In the form of equation 16, this corresponds to the matrix equation

$$\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} + \begin{bmatrix} -5 \\ -6 \end{bmatrix} = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}, \tag{22}$$

or the canonical system

$$\begin{aligned} a_1 \quad - 2f_1 - \quad f_2 &= -5, \\ a_2 - \quad f_1 - 2f_2 &= -6. \end{aligned} \tag{23}$$

We can also write equation 16 as a weighted sum of the vectors that contribute the columns of $\mathbf{I}$ and $-\mathbf{A}$,

$$a_1 \begin{bmatrix} 1 \\ 0 \end{bmatrix} + a_2 \begin{bmatrix} 0 \\ 1 \end{bmatrix} + f_1 \begin{bmatrix} -2 \\ -1 \end{bmatrix} + f_2 \begin{bmatrix} -1 \\ -2 \end{bmatrix} = \begin{bmatrix} -5 \\ -6 \end{bmatrix}. \tag{24}$$

From this last interpretation, it is easy to see that a complementary feasible solution to equation 16 exists if $\mathbf{b}$ is a nonnegative linear combination of the two remaining vectors when either $a_1 = a_2 = 0$, $a_1 = f_2 = 0$, $f_1 = a_2 = 0$ or $f_1 = f_2 = 0$. For example, if we set $a_1 = 0$ and $a_2 = 0$, then finding a solution reduces to finding $\mathbf{b}$ as a weighted sum of the columns of $-\mathbf{A}$, as illustrated in figure 11(a). The positive span of these two vectors is called a *complementary cone*.

Consider equation 20. Each variable a_i corresponds to the i^{th} column of $\mathbf{I}$, and each variable f_i corresponds to the i^{th} column of $-\mathbf{A}$. We call a_i and f_i each other's *complements*, since in a complementary solution, one or the other of them must be zero. Thus, a complementary solution of equation 16 is a weighted sum of n n-dimensional column vectors, where the i^{th} vector is the i^{th} column of either $\mathbf{I}$ or $-\mathbf{A}$. Each of the 2^n vector combinations for a given LCP corresponds to a complementary cone of the matrix $\mathbf{A}$. In a sense, the problem of finding a complementary feasible solution to an LCP is the same as finding a complementarity cone of $\mathbf{A}$ that contains $\mathbf{b}$. Figure 11(b) shows all complementary cones of the matrix

$$\mathbf{A} = \begin{bmatrix} 2 & -1 \\ 1 & 3 \end{bmatrix}. \tag{25}$$

Interestingly, since the union of the complementary cones of $\mathbf{A}$ spans $\mathbb{R}^2$, the LCP $(\mathbf{A}, \mathbf{b})$ has a solution for any choice of $\mathbf{b}$.

We conclude our introduction to LCPs with a simple but inefficient solution method, which Murty calls the *total enumeration method*. Let y_i be one of $\{a_i, f_i\}$, and let $\mathbf{y} = (y_1, \ldots, y_n)$. Let $\mathbf{Y}_i$ be the column of $\mathbf{I}$ or $-\mathbf{A}$ that corresponds to y_i. Let $\mathbf{Y}^r$ be the matrix formed from columns $\mathbf{Y}_1, \ldots, \mathbf{Y}_n$. Then, for each complementary vector $\mathbf{y}^r$, solve the linear program

$$\mathbf{Y}^r \mathbf{y}^r = \mathbf{b} \tag{26}$$

$$\mathbf{y}^r \geq \mathbf{0}. \tag{27}$$

Clearly, solving 2^n linear programs is not something we would do in practice, but this algorithm provides valuable intuition for understanding the LCP.

3.4 Lemke's complementary pivot method

In this section, we discuss Lemke's algorithm, which is commonly used to solve dynamics problems in robotics. We first present a high-level explanation in the terms described in the previous section, and then consider the algorithm in more detail. Our discussion is based on descriptions of the algorithm found in Cottle and Dantzig [15], Murty [49], and Cottle, Pang and Stone's volume on the LCP [16], which provides an excellent comprehensive treatment of the subject.

At the highest level, Lemke's algorithm is simply a series of steps between basic feasible solutions of 16. It begins with an *almost complementary* solution, that is, a solution which contains exactly one pair (a_i, f_i) such that $a_i f_i \neq 0$. The algorithm steps between solutions as long as the solution is almost complementary, following an *almost-complementary path*. If it reaches a solution that is complementary, then the algorithm terminates and returns this as the solution to the LCP. If the algorithm tries to take a step of infinite size, it terminates without finding a solution. For certain classes of the matrix $\mathbf{A}$, this implies that no complementary basic feasible solution exists.

The method finds an initial almost-complementary basic feasible solution as follows. For the general case, it is necessary to augment the original matrix $\mathbf{A}$ with a column of positive values. Typically, a column of ones, $\mathbf{e}_n$, is added. The vector $\mathbf{f}$ is augmented with an artificial variable f_0, as in equation

$$\begin{bmatrix} \mathbf{e} & \mathbf{A} \end{bmatrix} \begin{bmatrix} f_0 \\ \mathbf{f} \end{bmatrix} + \mathbf{b} = \mathbf{a}. \tag{28}$$

f_0 is increased until $\mathbf{a} \geq 0$ to find a starting solution.

The algorithm steps between solutions by selecting a nonbasic variable, called the *driving* variable, to increase by some amount. (Recall from section 3.3 that a nonbasic variable has a value of zero in a basic solution.) The driving variable is increased until it is *blocked* by some basic variable. We say a nonbasic variable is blocked if it reaches the point at which increasing it further would cause some basic variable to become negative. Since the algorithm maintains a feasible solution at all times, negative values are not allowed for any of the variables. Thus, the basic variable that has reached zero, called the *blocking* variable, is moved into the nonbasic set, and the driving variable is moved into the basic set. The process of exchanging variables between the basic and nonbasic sets is called a *pivot* operation, and may be characterized in several ways; see [17], Chapters 4 and 9. The *nondegeneracy assumption* by definition guarantees that at every step of the algorithm the blocking variable is unique. Techniques exist for dealing with degenerate matrices, and are addressed in section 4.9 of [16].

Because f_0 is increased to find an initial basic feasible solution, it is the initial driving variable. Subsequent driving variables are selected by the *complementary pivot rule*, which dictates that, after the initial step, the next driving variable is always the complement of the variable that has just left the basic set. This rule is the reason the algorithm is sometimes referred to as the complementary pivot method.

The algorithm may terminate in one of two ways. If f_0 leaves the basic set, then $(\mathbf{f}, \mathbf{a})$ is a complementary basic feasible solution, and the algorithm terminates with this solution to the LCP. Alternatively, the algorithm may attempt to take a step of infinite size. In this case, the algorithm terminates without finding a solution. Depending on the class of matrix $\mathbf{A}$, this may indicate that a solution exists but the algorithm was unable to find it, or that no solution exists.

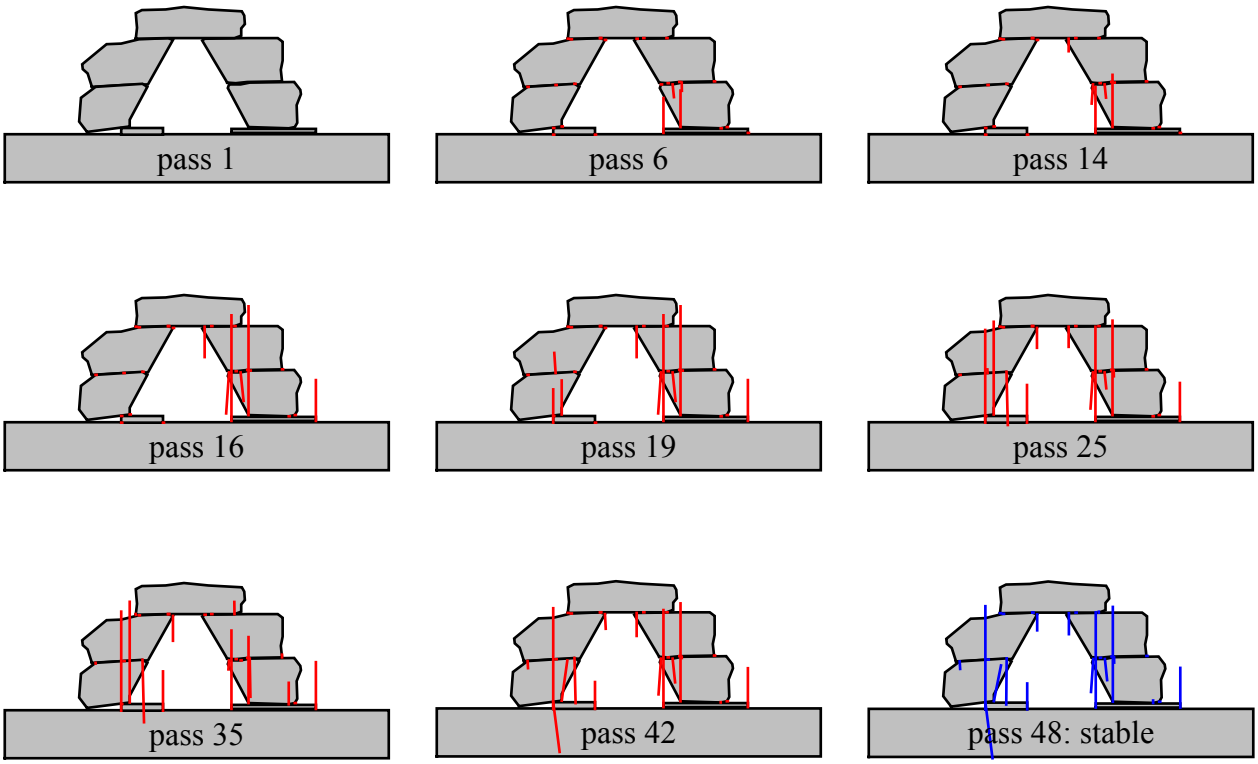


Figure 12: Intermediate solutions in Dantzig's method illustrated as force magnitudes at contact points in structure.

3.5 Dantzig's principle pivoting method

In this section, we describe the principle pivoting method first in the terms of Cottle and Dantzig [15], and draw an analogy between it and Lemke's algorithm. We then give a physical interpretation of the algorithm in terms of the system dynamics, and look in greater detail at the method in the terms of Baraff, who proposed this as fast algorithm for computing contact forces for his formulation of the system dynamics in [6]. We focus on this method in greater detail than on Lemke because our primary result for computation reuse is based on modifications to this method.

In contrast to Lemke's complementary pivot method, which steps between basic feasible solutions, Dantzig's principle pivoting method steps between complementary basic solutions. It progresses by means of *major* and *minor cycles*. The goal of a major cycle is to achieve feasibility for a single pair of complementary variables (f_i, a_i). During the cycle, the complementarity condition is relaxed for these variables, so the intermediate solutions are points along an almost complementary path. Some number of minor cycles occur within the major cycle, to ensure that feasibility and complementary conditions are maintained for all variables that have already attained feasibility. At the end of the major cycle, the pair is feasible and complementarity. The algorithm continues until a complementary feasible solution is attained.

A starting complementary solution is found by initializing $(\mathbf{f}, \mathbf{a})$ with the complementary solution $(\mathbf{0}, \mathbf{b})$. If the problem is nontrivial, there exists an index i such that $a_i < 0$, and f_i is selected as the driving variable. There may be many values of i that satisfy this condition; the driving variable is selected arbitrarily. It is this arbitrary selection that makes computation reuse possible in our modification of the algorithm.

Once a driving variable f_i has been selected, a major cycle begins, with the goal of increasing f_i until a_i reaches zero. Within a major cycle there may be some finite number of minor cycles. The driving variable f_i (which is nonbasic and begins with a value of zero) is increased until it is blocked by some basic variable. The blocking variable is then transfered to the nonbasic set, and its complement enters the basic set. Because $\mathbf{A}$ has positive principle minors, this exchange of the blocking variable with its complement permits the further increase of f_i (by [15], theorem 10). A

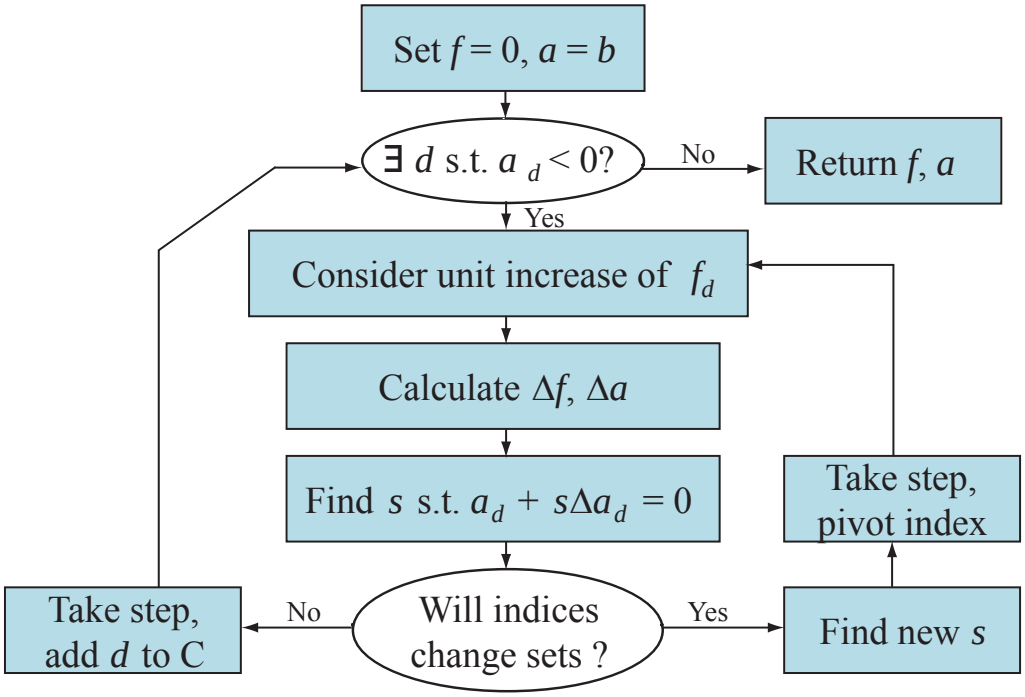


Figure 13: Block diagram for Baraff's implementation of Dantzig's algorithm.

new minor cycle begins, and f_i is increased again. The major cycle terminates when a_i, the complement of the driving variable, reaches zero. f_i enters the basic set, a_i moves to the nonbasic set, and a new major cycle begins. The process terminates when no negative variables remain in $\mathbf{a}$.

A major cycle of Dantzig's method is in some ways analogous to Lemke's method as a whole. An initial nonbasic driving variable is chosen and is increased until it is blocked by some basic variable, at which point a pivot operation occurs. The difference here is that, instead of exchanging the blocking variable with the driving variable, the blocking variable is exchanged with its own complement, and the driving variable is the same over the course of the entire major cycle. However, much like Lemke's method, the algorithm proceeds along an almost-complementary path until a complementary solution is achieved.

3.5.1 Physical interpretation

Baraff's algorithm for solving the system dynamics in [6] is equivalent to Dantzig's method in the frictionless case. We can use his model to give an interpretation of what is happening physically as the algorithm progresses.

As explained in section 2, Baraff is concerned with finding a vector of contact force magnitudes, $\mathbf{f}$, that satisfies the system dynamics equations. For a given structure, we set all the contact forces equal to zero, and then consider the result that doing so has on the contact accelerations. Clearly, the objects in the structure will accelerate into each other because of the gravitational force on them. We then consider each contact point in sequence, and attempt to increase the magnitude of the contact force at that point until the relative acceleration between the two objects is zero. Since the force magnitude variable we increase is the driving variable, for our physical explanation we will call this the driving force. If, in the process of increasing the driving force, some contact force between two other blocks is in danger of becoming aphysical, *i.e.* becomes zero and would be an attractive force if the driving force were to continue to increase, we exchange force and acceleration at this contact point and, instead of reducing the contact force at this

point, we increase the relative acceleration between the two blocks. Alternatively, if some contact acceleration that was decreasing as a result of our increase to the driving force would become aphysical, *i.e.* would allow two contacting bodies to accelerate toward each other causing interpenetration, then we stop reducing the acceleration at this contact point and instead increase the magnitude of the contact force. We continue this process for every contact force in the structure until we have found a solution such that the structure is at rest or contains objects are accelerating away from each other.

Figure 12 illustrates the solution process for Dantzig's method by showing the intermediate force magnitudes calculated during the solution of the LCP for this physical structure. Negative values are not drawn.

3.5.2 Baraff's implementation

This process is implemented as illustrated in figure 13. We set the force vector $\mathbf{f}$ to zero, and consider the resulting acceleration vector $\mathbf{a}$. If $(\mathbf{f}, \mathbf{a})$ is feasible, then the procedure terminates with this solution. Otherwise, we select a driving variable f_d such that $a_d < 0$. In order to determine how much we can increase f_d before it is blocked, we consider a unit increase to f_d, and examine the effect of this increase on the forces and accelerations at previously considered contact points.

As we consider each index i from 1 to n, once the pair (f_i, a_i) is feasible, we place i in an *index set* to determine which variable of the pair is basic and which is nonbasic. We put i in the set C if f_i is basic, or i in NC if a_i is basic. Recall that basic variables are dependent variables, and in a basic solution, nonbasic variables have a value of zero. Baraff picked C to indicate "clamped" contact points where $f_i \geq 0$, and NC to indicate "not-clamped", or separating, contact points where $a_i \geq 0$.

The index sets C and NC help us find $\Delta\mathbf{f}$ and $\Delta\mathbf{a}$, the changes to vectors $\mathbf{f}$ and $\mathbf{a}$ in a given minor cycle. For a unit increase of f_d, we maintain $f_i = 0$ for all $i \in NC$ and $a_i = 0$ for all $i \in C$. Accordingly, $\Delta f_d = 1$, $\Delta f_i = 0$ for all $i \in NC$, so we can arrange the rows of $\Delta\mathbf{f}$ such that

$$\Delta\mathbf{f} = \begin{bmatrix} \mathbf{x} \\ \mathbf{0} \\ 1 \end{bmatrix} \begin{matrix} \leftarrow i \in C \\ \leftarrow i \in NC \\ \leftarrow i = d \end{matrix} \tag{29}$$

The above equation is annotated to show that indices $i \in C$ are in the first row, $i \in NC$ are in the second, and the index of the driving variable is in the final row. We can partition $\mathbf{A}$ similarly,

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} & \mathbf{v}_1 \\ \mathbf{A}_{12}^T & \mathbf{A}_{22} & \mathbf{v}_2 \\ \mathbf{v}_1^T & \mathbf{v}_2^T & \alpha \end{bmatrix}. \tag{30}$$

$\Delta\mathbf{f}$ and $\Delta\mathbf{a}$ are related by

$$\Delta\mathbf{a} = \mathbf{A}(\mathbf{f} + \Delta\mathbf{f}) + \mathbf{b} - (\mathbf{A}\mathbf{f} + \mathbf{b}) = \mathbf{A}\Delta\mathbf{f}, \tag{31}$$

so $\Delta\mathbf{a}$ may also be partitioned as

$$\Delta\mathbf{a} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} & \mathbf{v}_1 \\ \mathbf{A}_{12}^T & \mathbf{A}_{22} & \mathbf{v}_2 \\ \mathbf{v}_1^T & \mathbf{v}_2^T & \alpha \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{0} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{11}\mathbf{x} + \mathbf{v}_1 \\ \mathbf{A}_{12}^T\mathbf{x} + \mathbf{v}_2 \\ \mathbf{v}_1^T\mathbf{x} + \alpha \end{bmatrix} \tag{32}$$

Since we seek to find $\Delta\mathbf{f}$ such that $\Delta a_i = 0$ for all $i \in C$, the top row of $\Delta\mathbf{a}$ must be zero. Thus, $\mathbf{x}$ must satisfy

$$\mathbf{A}_{11}\mathbf{x} = -\mathbf{v}_1. \tag{33}$$

Once $\mathbf{x}$ is known, $\Delta\mathbf{f}$ and $\Delta\mathbf{a}$ are easily calculated from

$$\Delta\mathbf{f} = \begin{bmatrix} \mathbf{x} \\ \mathbf{0} \\ 1 \end{bmatrix}, \qquad \text{and} \qquad \Delta\mathbf{a} = \mathbf{A}\Delta\mathbf{f}. \tag{34}$$

Baraff shows that equation 33 always has a solution.

To complete the major cycle, we must find s such that

$$a_d + s\Delta a_d = 0, \tag{35}$$

however, s must also satisfy

$$\mathbf{f} + s\Delta\mathbf{f} \geq \mathbf{0}, \qquad \text{and} \qquad \mathbf{a} + s\Delta\mathbf{a} \geq \mathbf{0}. \tag{36}$$

The greatest value of s that satisfies equation 36 is found in linear time by iterating through all previously considered contact points. If s can be large enough to satisfy equation 35, the major cycle ends, otherwise, another minor cycle begins.

3.6 Pivot rules in Lemke and Dantzig

Essentially, we can reuse computation in Dantzig's algorithm because the choice of pivot element at the start of a major cycle is arbitrary. In contrast, Lemke's algorithm completely specifies each pivot element with its complementary pivot rule, and the proof of correctness of the algorithm is based on this pivot choice [15]. We do not give a formal proof, but we conjecture that it is not possible to reuse computation in Lemke's algorithm. However, optimal methods for selecting pivot elements are not completely known theoretically [56], so it is possible that an algorithm similar to Lemke's could be devised for which computation reuse is possible for the same larger class of matrices. Developing such an algorithm is left as an open problem.

4 Computation reuse in the linear complementarity problem

In this section, we present the first major results of this thesis. First, in sections 4.1 and 4.2 we present a theoretical reuse result for LCPs. Dantzig's algorithm, when applicable, will find a solution to the linear complementarity problem corresponding to an assembly with n contacts in n or fewer major cycles. The actual number of major cycles depends on the values of $\mathbf{A}$ and $\mathbf{b}$ in the problem, but for the LCPs corresponding to the physical structures we have considered, Dantzig's algorithm usually takes all or very close to n major cycles. We show that Dantzig's algorithm will find a solution in $n - k$ or fewer major cycles if the algorithm is initialized with a solution to the dynamics problem for a subassembly with k internal contacts. In the assembly planning application we consider in section 4.3, k is typically $n - 2$, so the time savings from reusing computation is substantial.

We then discuss several applications of this result, first to the stability problem, and then to dynamic simulation. Specifically, in section 4.3 we use this result to generate a *stable disassembly sequence*, that is, an order in which we can remove every object from the structure without causing it to collapse under gravity after any step. We show that, by reusing computation in this way in the dynamics formulation, we are able to find a stable disassembly sequence in a constant number of major cycles, and in less time than if we use a simpler statics formulation, determining stability by solving a linear program. In section 4.4, we use the LCP result to improve the performance time of a dynamic simulation in which the structure being simulated shares a common geometry between time steps. If the unmoved substructure has k internal contacts, then we can solve the dynamics formulation for the next frame in $n - k$ or fewer major cycles using Dantzig's algorithm modified with computation reuse.

4.1 Computation reuse

To solve an LCP $(\mathbf{A}, \mathbf{b})$, Dantzig's principle pivoting method may execute as many as n major cycles. This is shown in [15], by the argument that at the end of each major cycle, the number of negative elements in the vector $\mathbf{a}$ is reduced by one, so the method can make no more than n major cycles. We show in this section that if we initialize the algorithm with a solution to an LCP corresponding to a subassembly, we reduce the maximum number of major cycles the algorithm can make.

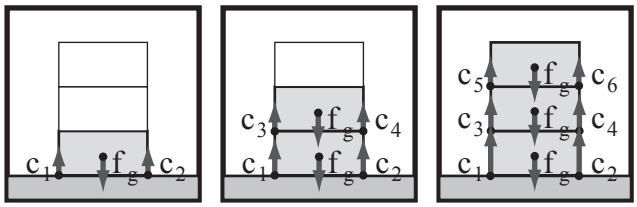


Figure 14: Sequential computation of contact forces for subassemblies. c_i indicates the ith contact force and f_g the gravitational force. The length of an arrow shows the relative magnitude of the force it represents.

Theorem 1 *Let $(\mathbf{A}, \mathbf{b})$ be an LCP, let $\mathbf{A}_{11}$ be any $k \times k$ principle submatrix of $\mathbf{A}$, and let $\mathbf{b}_1$ be a k-dimensional subvector of $\mathbf{b}$ with the same rows removed. If $(\mathbf{x}, \mathbf{y})$ is a complementary feasible solution to the LCP $(\mathbf{A}_{11}, \mathbf{b}_1)$, then initializing Dantzig's algorithm with the complementary solution $([\ \mathbf{x}\ \mathbf{0}\ ]^T, [\ \mathbf{y}\ \mathbf{A}_{21}\mathbf{x} + \mathbf{b}_2\ ]^T)$ will allow it to find a complementary feasible solution for the LCP $(\mathbf{A}, \mathbf{b})$ in at most $n - k$ major cycles.*

Proof: Observe that if $(\mathbf{x}, \mathbf{y})$ is a complementary feasible solution to the LCP $(\mathbf{A}_{11}, \mathbf{b}_1)$, then there exists $\mathbf{a}$ such that $([\ \mathbf{x}\ \mathbf{0}\ ]^T, \mathbf{a})$ is a complementary solution to $(\mathbf{A}, \mathbf{b})$. Specifically, since the order in which we number contacts is arbitrary, we can arrange $\mathbf{A}$ and $\mathbf{b}$ such that $\mathbf{A}_{11}$ is the upper left $k \times k$ principle submatrix of $\mathbf{A}$, and $\mathbf{b}_1$ is the upper k-dimensional subvector of $\mathbf{b}$. Thus, we may partition $\mathbf{A}$ and $\mathbf{b}$ as in the equation

$$\begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{0} \end{bmatrix} + \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{y} \\ \mathbf{A}_{21}\mathbf{x} + \mathbf{b}_2 \end{bmatrix}. \tag{37}$$

Since $(\mathbf{x}, \mathbf{y})$ is complementary, $(\mathbf{f}, \mathbf{a})$, where $\mathbf{f} = [\ \mathbf{x}\ \mathbf{0}\ ]^T, \mathbf{a} = [\ \mathbf{y}\ \mathbf{A}_{21}\mathbf{x} + \mathbf{b}_2\ ]^T$, is a complementary solution for $(\mathbf{A}, \mathbf{b})$.

The argument that the maximum number of major cycles Dantzig's algorithm can make if initialized with this complementary solution is $n - k$ follows that in [15]. Each major cycle reduces the number of negative variables in $\mathbf{a}$ by at least one. Since $\mathbf{y}$ is feasible, all negative variables in $\mathbf{a}$ are contained in the $(n - k)$-dimensional subvector $\mathbf{A}_{21}\mathbf{x} + \mathbf{b}_2$. Thus, the algorithm can make no more than $n - k$ major cycles. ■

4.2 Reuse for physical structures

Figure 14 gives some intuition for applying Dantzig's algorithm to a physical system. In the left frame, the contact forces on the bottom block are computed without considering the force contributions of the other blocks. In the center frame, the second block is added, and the contact forces on the first block are adjusted; contact forces on the second block are computed. In the right frame, the final block is added, and the contact forces associated with the first two blocks are adjusted.

The optimization described in the previous section can be used to reduce the computation required to find contact forces in assemblies of rigid bodies if a complementary feasible solution to a substructure LCP is known. Specifically, we show that for the LCP $(\mathbf{A}, \mathbf{b})$ from Baraff's model of the system dynamics, described in equation 12, $(\mathbf{A}_{11}, \mathbf{b}_1)$ is exactly the LCP corresponding to a substructure.

In equation 12, rows of $\mathbf{J}$ correspond to contacts in the structure and columns correspond to bodies, rows and columns of $\mathbf{M}$ correspond to bodies, as do rows of $\mathbf{F}^{\text{ext}}$. Thus, for a superstructure, we can partition these terms as

$$\mathbf{J} = \begin{bmatrix} \mathbf{J}_{11} & \mathbf{J}_{12} \\ \mathbf{J}_{21} & \mathbf{J}_{22} \end{bmatrix}, \qquad \mathbf{M}^{-1} = \begin{bmatrix} \mathbf{M}_{11}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{M}_{22}^{-1} \end{bmatrix}, \qquad \mathbf{F}^{\text{ext}} = \begin{bmatrix} \mathbf{F}_1^{\text{ext}} \\ \mathbf{F}_2^{\text{ext}} \end{bmatrix}, \tag{38}$$

where the top and left partitions correspond to elements present in the superstructure but not in the substructure. Then,

$$\begin{aligned} \mathbf{A}_{11} &= \mathbf{J}_{11}\mathbf{M}_{11}^{-1}\mathbf{J}_{11}^T + \mathbf{J}_{12}\mathbf{M}_{22}^{-1}\mathbf{J}_{12}^T, \\ \mathbf{b}_1 &= \mathbf{J}_{11}\mathbf{M}_{11}^{-1}\mathbf{F}_1 + \mathbf{J}_{12}\mathbf{M}_{22}^{-1}\mathbf{F}_2. \end{aligned} \tag{39}$$

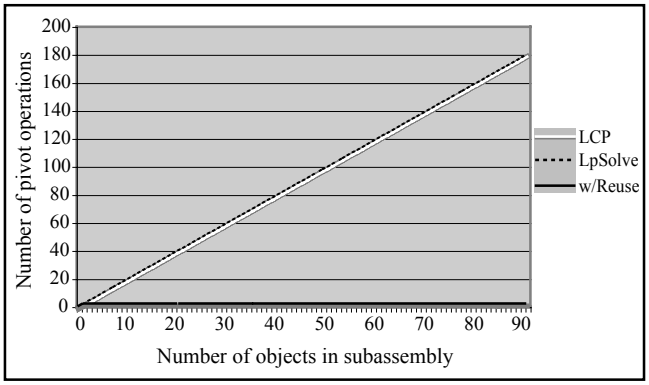


Figure 15: Number of pivots required by a single stability test, by the size of the subassembly tested, for a 91-block pyramid.

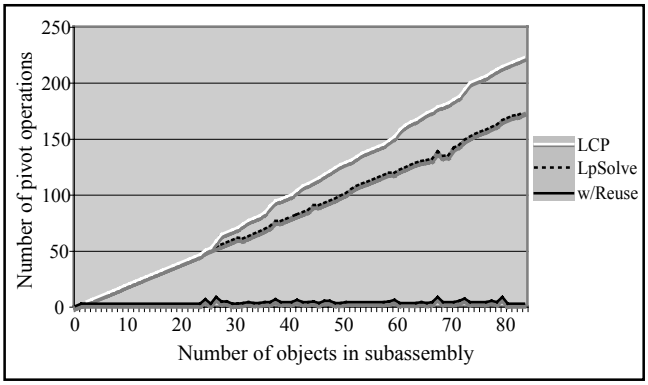


Figure 16: Number of pivots required by a single stability test, by the size of the subassembly tested, for the 84-block randomly generated structure show in figure 19(b).

However, $\mathbf{J}_{12} = \mathbf{0}$, since no contacts present in the substructure are contacts between objects that are not in the substructure. Thus equation 39 reduces to

$$\begin{aligned} \mathbf{A}_{11} &= \mathbf{J}_{11}\mathbf{M}_{11}^{-1}\mathbf{J}_{11}^{T}, \\ \mathbf{b}_1 &= \mathbf{J}_{11}\mathbf{M}_{11}^{-1}\mathbf{F}_1, \end{aligned} \tag{40}$$

the exact formulation of the substructure LCP.

4.3 Application: assembly planning

A fundamental problem in mechanical assembly is to find an order in which a product can be assembled or disassembled. We consider the problem of finding a *stable disassembly sequence*, that is, an order in which we can remove every object from the structure without causing it to collapse under gravity after any step. Structures with substructures in identical configurations arise naturally in this problem. Researchers have considered reusing similar geometries for moveability [29, 73], but, to our knowledge, not for determining stability. In automated assembly sequencing, it is typical to determine stability by solving a linear program (LP) corresponding to the static force-balance equations [59]. We show that our LCP stability test with computation reuse outperforms this approach, even though we expect system dynamics to be more complicated than statics, as illustrated by the difference between LpSolve and Dantzig LCP columns in table 2.

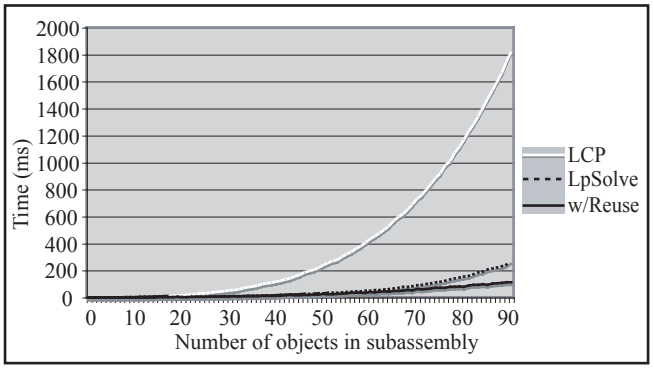


Figure 17: Time taken for a single stability test, by the size of the subassembly tested, for a 91-block pyramid.

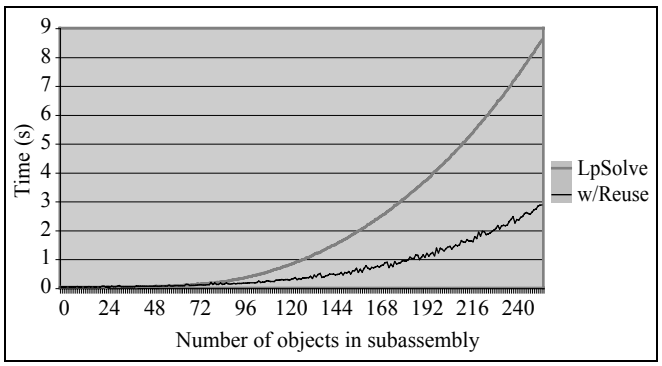


Figure 18: Time taken for a single stability test, by the size of the subassembly tested, for a 253-block pyramid.

A basic approach for finding a stable disassembly sequence is as follows. Consider the set of all objects in a structure. Collect all possible subsets into a *disassembly graph*, as shown in figure 3. Search the graph, testing the stability of each node as it is visited. Any path of stable nodes from start to goal is a stable disassembly sequence.

Many researchers including [20, 28, 73] have considered the problem of improving the performance of this algorithm by reducing the number of nodes visited; we focus on reducing the amount of time spent testing the stability of a node. We can test stability by computing a solution for the contact forces, and then verifying that body accelerations are zero. As discussed in section 2, if there is friction, this test is only a necessary condition for stability; computing *strong* stability is beyond the scope of this thesis.

Since we add a single block at a time and only consider successors to stable nodes, we can use the complementary feasible solution from any parent of the current node to initialize the stability test, reducing the maximum number of major cycles in the stability test from n to a small constant.

To verify the feasibility and correctness of our optimizations, we have developed an efficient unstacking planner for planar systems with static friction. Computation reuse has been implemented for assemblies with and without friction. The planner can be configured to find a stable unstacking sequence using one of three stability tests: by finding a solution to the system statics, solving the system dynamics using Dantzig's algorithm, or by solving the system dynamics using Dantzig's algorithm with our modifications for computation reuse. The system statics stability test was implemented with `lp_solve` – a fast, open-source linear program solver written in C, that uses the revised simplex method [9] – to determine the feasibility of a structure's LP.

Figures 15 and 16 show the differences in the number of pivot operations between the three algorithms for a single stability test of a frictionless 91-block structure, and the frictionless 84-block randomly generated structure show in figure 19(b). Figure 17 shows the time difference between the three algorithms for the 91-block pyramid. Figure 18 shows the time difference between the linear programming approach and the LCP approach with computation reuse

Table 1: Total number of pivots by algorithm.

Structure	Blocks	LpSolve	Dantzig LCP	LCP + Reuse
Random	10	102	127	13
Column	50	2352	2352	96
Column	80	6162	6162	156
Random	84	11,975	15,136	420
Pyramid	91	8190	8190	180

Table 2: Average time by algorithm, measured over 100 executions.

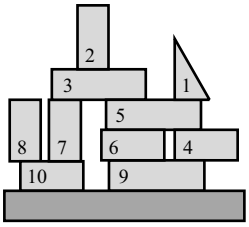
Structure	Blocks	LpSolve	Dantzig LCP	LCP + Reuse
Random	10	0.017 s	0.024 s	0.010 s
Column	50	0.483 s	2.987 s	0.469 s
Column	80	2.880 s	20.997 s	2.005 s
Random	84	9.396 s	72.099 s	5.532 s
Pyramid	91	5.260 s	38.825 s	3.074 s

implemented, for a single stability test of a frictionless 253-block pyramid. In each case, a depth-first search strategy was used to explore the assembly graph. Our implementation of the LCP algorithm with computation reuse, written in Java, consistently outperformed the `lp_solve` implementation, as shown in table 2.

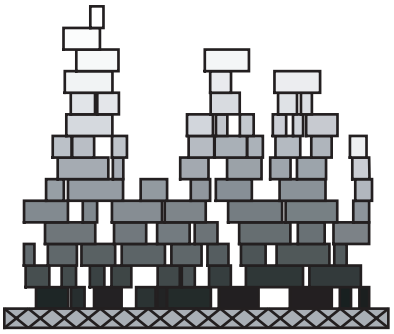
We have not tested the stable disassembly sequence algorithm using Lemke's method as the stability test, so we do not include results for Lemke's method in our comparison here. However, we note that Lemke's method is similar to the simplex method and Dantzig's method for LCPs, in that it is also a pivoting method, and has an exponential running time in the worst case, but an expected running time that is polynomial in n. We expect Lemke to require roughly the same number of pivot operations as simplex or Dantzig's LCP method, thus, we would expect its running time to be similar to that of LpSolve or Dantzig's method without computation reuse, depending on the implementation.

We tested the different algorithms on a variety of examples, including columns, pyramids, and randomly-generated structures, such as the one shown in figure 19(b). More detailed discussion of early examples can be found in [33]. For large structures, memory is the limiting factor. With 256 MB allocated to the Java Virtual Machine, and testing stability using Dantzig's algorithm with computation reuse, a stable disassembly sequence can be found for a maximum-sized structure of 280 blocks. This is because, for optimal time-efficiency, we store the matrix $\mathbf{A}$ for every node in the path from the root to the current node. We conjecture that there is a time/space efficiency tradeoff that can be made to reduce the number of nodes at which we store the $\mathbf{A}$ matrix and still substantially improve the performance time of the total number of stability tests the algorithm makes; however, finding the optimal number of matrices stored is left as an open problem. Table 1 gives the total number of pivots in both major and minor cycles of the algorithm for a few example problems. Table 2 presents the run-time of the algorithm for the same examples. All tests were run on a 1.5 GHz PowerBook G4.

In addition to the unstacking platform, we have developed software for generating structures in different ways. The first parses an `svg` file that contains a structure drawn in Adobe Illustrator and converts it to a format our planner can use. The second generates a structure at random using the following procedure. Starting with an empty world, pick a random offset on the first row and place a block there. Test the stability of the resulting structure. If it is unstable, remove the last block that was added and continue. Repeatedly pick rows and offsets at random, placing blocks and testing the stability of the assembly, until the structure is a prespecified number of rows high. The structure in figure 19(b) was generated with this algorithm.



(a) A simple structure, labeled with an unstacking order.



(b) A randomly-generated assembly, where unstacking order is indicated by block color. Observe that blocks in the column on the far right must be removed before other blocks at higher levels in the structure.

Figure 19: Several assemblies and their unstacking sequences.

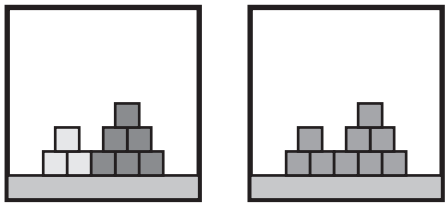


Figure 20: Non-intersecting stable subassemblies.

4.3.1 Graph search

In addition to finding ways to reuse computation in the test for stability, the basic unstacking algorithm can be improved by finding better paths through the disassembly graph. We have made steps toward improving the graph search by identifying strategies that may allow us to test fewer nodes in the graph by stability properties. While this has not been a primary focus of this thesis research, it is of interest because heuristics for culling the search graph are known based on kinematic constraints, as discussed in section 2.4, but not based on stability. We have not added these methods to our software implementation.

Culling rules. Since we are interested only in complete paths from root to leaf in the assembly graph, some culling of nodes may be possible during the search.

1. If a node is unstable, cut all of its edges.
2. If there is a stable path to a node from the root, cut all of the node's entrant edges.
3. If there is a stable path to a node from the leaf, cut all of the node's exit edges.

If a subgraph has no entrant edges or exit edges, the stability of the nodes in the subgraph need not be calculated. We have not determined whether efficient ordering of the search could be used to remove significant subgraphs from consideration.

Union of subassemblies. Intuitively, if two subassemblies are weakly or strongly stable and do not touch, their union is also weakly or strongly stable. In fact, even if the stable subassemblies touch, their union is guaranteed to be at least weakly stable.

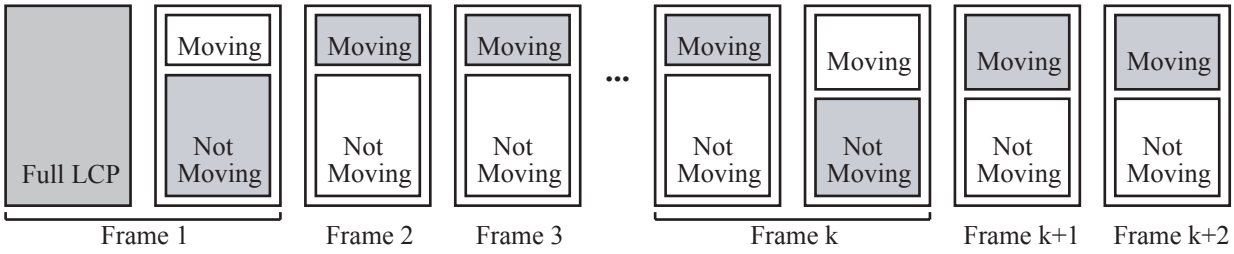


Figure 21: Illustration of how an LCP solution may be reused in dynamic simulation. Shaded areas represent the portion of the LCP that must be evaluated at each step. In frame k, an element that was previously motionless begins to move, so we must solve the LCP corresponding to the new motionless set and store it for reuse in subsequent frames.

Theorem 2 (Union of subassemblies) *The union of non-intersecting weakly-stable subassemblies is a weakly-stable assembly.*

Proof: Consider the contact graph for the assembly. Each subassembly is stable. Set the force magnitudes at contacts connecting subassemblies to zero. ■

Heuristics, search order, and existence. Since all successful stacking sequences are of length n, depth-first search is an obvious choice for searching the disassembly graph. Other strategies may be more effective for some types of structures. Search algorithms that give preference to stacking from the ground up also seem to be more likely to succeed quickly.

We expect artificial structures to have a stable unstacking sequence, but the set of structures that are stably unstackable may be small. Whether we can determine the existence of a stacking sequence for a structure more efficiently than we can construct the sequence is an open question.

We expect squat structures will be more likely to be stable than tall spires.

4.4 Application: dynamic simulation

In this section, we look at how the computation reuse result from sections 4.1 and 4.2 can be applied to the problem of dynamic simulation. During simulation, it is often the case that a subset of contacting rigid bodies remains motionless for many time steps. Reusing the solution for the persistent substructure allows us to optimize the contact force calculation for the simulation as described in the previous section. Figure 21 illustrates the concept behind the algorithm. We compute the dynamics to identify the set of motionless bodies, re-compute the dynamics of that substructure as a partial solution, and use this partial solution as a starting point to compute the dynamics of the complete structure in successive time steps. If the set of motionless bodies changes, we compute the partial solution corresponding to the new motionless set, and use it until the motionless set changes again. This may be written

```
SIMULATE(world)
1  prevSoln ← ∅
2  prevMotionless ← ∅
3  time ← 0
4  while (time++ < SIMULATIONTIME)
5       do SOLVELCP(world, prevSoln)
6          motionless ← GETMOTIONLESSBODIES(world)
7          if (prevMotionless ≠ motionless)
8            then prevSoln ← SOLVELCP(motionless, ∅)
9                 prevMotionless ← motionless
```

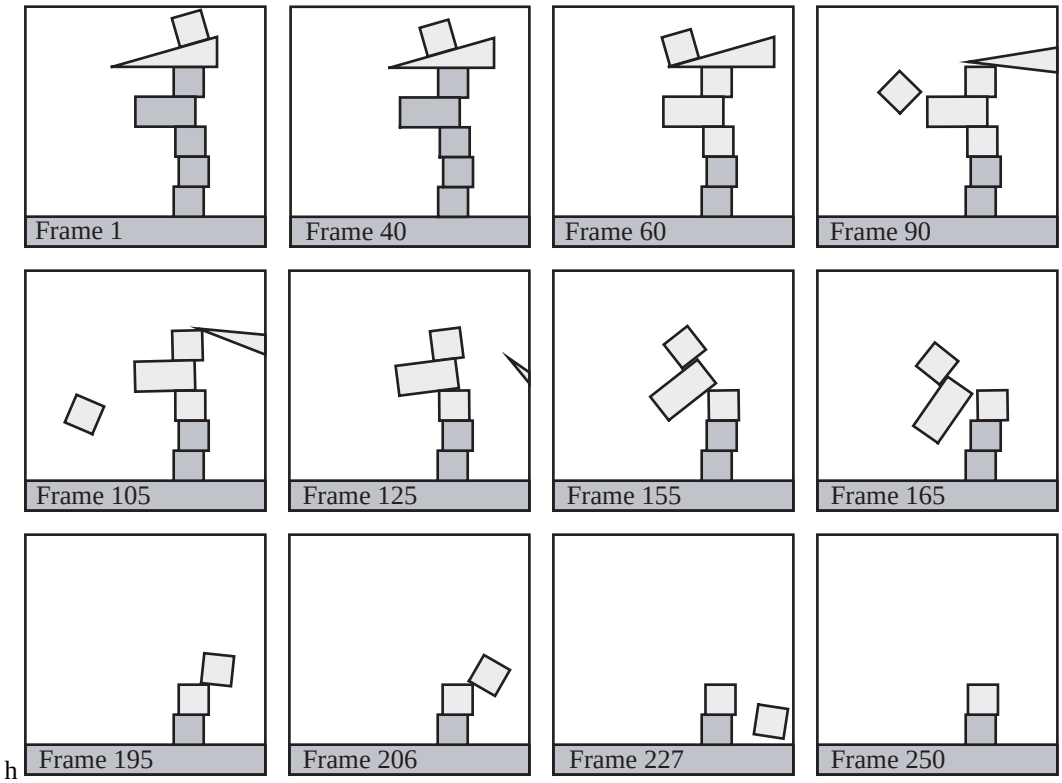


Figure 22: A column falling apart. Dark gray blocks are motionless. The motionless set changes at frames 60 and 195.

We might further improve the performance of this algorithm at line 8, by reusing the solution from the prior frame corresponding to the elements of the new motionless set, instead of solving the new LCP from scratch. It is well-known that starting a linear program with a solution to a similar problem often reduces the time to solve it, but we have not focused on determining whether the same is true for LCPs.

We have implemented a planar dynamic simulator with friction based on the design described by Baraff and Witkin in their 2001 SIGGRAPH Course Notes [8]. The simulator uses the same components as the disassembly planner to read in a structure's description, before simulating the system dynamics. At each time step, we apply the external and contact forces to find the acceleration of each body, and then integrate accelerations to find the location and velocity of the bodies in the next time step. The algorithm described above is implemented to exploit computation reuse. In the frictionless case, the algorithm reduces to Dantzig's principal pivoting method. For friction, we use Baraff's modification to Dantzig's algorithm. This formulation is not a true LCP, as it also includes a number of auxiliary conditions in addition to the standard complementary and feasibility conditions of an LCP [6]. Baraff is unable to prove that this algorithm converges, but is considered to be reliable in practice.

The SIMULATE function with computation reuse is also applicable to Baraff's three-dimensional model, and to three-dimensional models that employ linearized friction cones, where $\mathbf{A}$ is a P-matrix or is PSD.

For the example in figure 22, simulated with a small coefficient of friction, our optimized algorithm reduced the time spent solving LCPs from 2.2 s to 1.2 s, a reduction of 45%. For those frames where five of the seven blocks are motionless, the average savings from computation reuse was 65%.

We expect the simulation reuse algorithm to be fastest when the motionless structure is large and when we are able to reuse a solution across many frames, as is the case when we use a small integration step. Thus, this algorithm allows a more fine-grained time step, and we expect it to be particularly useful for applications where a highly detailed simulation is required, such as mechanism analysis.

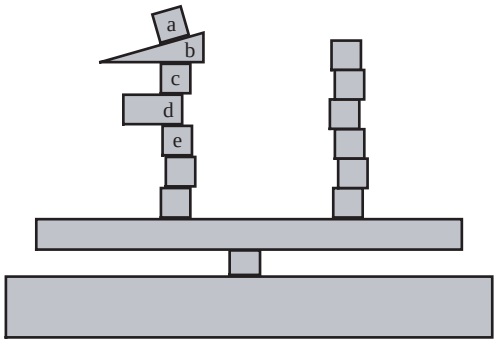


Figure 23: Three different motionless sets appear in this simulation: first only a and b are in motion, then c, d, and e move as well, then all blocks fall.

Although reducing the number of major cycles in Dantzig's algorithm usually improves the overall running time of the algorithm, this is not always the case. The number of minor cycles within a major cycle depends on the order in which contacts are considered. We have observed that in some cases, the contact ordering imposed by the computation reuse algorithm causes major cycles to take more time. In the example in figure 23, there are three simulation segments. For the first 41 frames, only blocks a and b are in motion. In the next 46 frames, c, d, and e are in motion as well. For the remainder of the simulation, all blocks except for ground are in motion. LCPs in the second segment take nearly four times as long to compute with computation reuse, in spite of the fact that the algorithm makes only 22 major cycles instead of 60. This is likely because the average number of minor cycles within a major cycle is high for the particular LCP we are solving at each step.

Methods for determining when reusing computation is less efficient are left as an open problem. Another open problem is how to determine an optimimal sensitivity level for designating objects as being in motion.

4.5 Open Problems

Throughout this section, we have noted where other researchers might extend our results. The most notable open problems involve extending our computation reuse results to complementarity problem algorithms for larger classes of matrices, to nonlinear program solvers, and to addressing the questions that remain in the assembly stability problem, many of which were outlined in section 5.6.

In this section, we list several additional open problems we have identified regarding computation reuse in statics and dynamics.

4.5.1 Computation reuse for linear programs

In addition to considering system dynamics, we might try to reuse computation in the statics formulation. One idea is to use an approach similar to the one used for the LCP formulation. Consider the LP for a subassembly of one block. Solve for the contact force magnitudes using the simplex method. Add an object to the structure, and add the necessary rows and columns to the LP, applying to them the affine transformation that would have been applied had they been present in the LP from the beginning of the problem.

Dual form of the system statics. Given a linear program of the form

$$\max \mathbf{cx}, \qquad \mathbf{Ax} = \mathbf{b}, \qquad \mathbf{x} \geq \mathbf{0}, \tag{41}$$

its dual is

$$\min \mathbf{by}, \qquad \mathbf{A}^T\mathbf{y} = \mathbf{c}, \qquad \mathbf{y} \text{ unrestricted.} \tag{42}$$

Equation 41 is known as the primal problem.

By the duality theorem of linear programming [17], if both the primal and dual have solutions, then they both have optimal solutions, and these solutions are equivalent. A consequence of this theorem is that if the dual is feasible but unbounded, then the primal does not have a feasible solution.

We might use this to exploit computation reuse for solving the system statics. The dual of system given in equation 8 is

$$\mathbf{J}\mathbf{y} = \mathbf{c}, \qquad \mathbf{y} \text{ unrestricted}, \tag{43}$$

where $\mathbf{c}$ is an objective function whose values we are unconcerned with. Presumably, we can choose $\mathbf{c}$ to have any value. If we add objects to a structure, $\mathbf{J}$ has the form,

$$\mathbf{J} = \begin{bmatrix} \mathbf{J}_{11} & \mathbf{0} \\ \mathbf{J}_{21} & \mathbf{J}_{22} \end{bmatrix} \tag{44}$$

where $\mathbf{J}_{11}$ is the Jacobian of the assembly before the object were added. If we view this in terms of the geometric interpretation of the simplex method, *i.e.* traversing the edges of a simplex in n-dimensional space until an optimal solution is reached, then the variables added in column 2 simply extrude the simplex in these dimensions. If the dual was bounded for the linear program

$$\min \mathbf{b}_1\mathbf{y}_1, \qquad \mathbf{J}_{11}\mathbf{y}_1 = \mathbf{c}, \qquad \mathbf{y} \text{ unrestricted}, \tag{45}$$

then it seems likely that we should be able to determine the boundedness in terms of the variables in $\mathbf{y}_2$ in a reduced amount of time. Determining whether this is the case is left as an open problem.

Solving an LP as an LCP. Alternatively, we might try to convert the LP to an LCP and solve the LCP in a manner that reuses computation. The following is a discussion of how to convert an LP to an LCP, as presented by Cottle in [15]. Consider an LP in the primal-dual form. In the primal LP, the goal is to find $\mathbf{x}$ such that the objective function $\mathbf{z}_p$ is minimized:

$$\mathbf{A}\mathbf{x} \geq \mathbf{b}, \qquad \mathbf{x} \geq \mathbf{0}, \qquad \mathbf{z}_p = \mathbf{c}\mathbf{x}. \tag{46}$$

In the dual LP, the goal is to find $\mathbf{y}$ such that the objective function $\mathbf{z}_d$ is maximized:

$$\mathbf{y}\mathbf{A} \leq \mathbf{c}, \qquad \mathbf{y} \geq \mathbf{0}, \qquad \mathbf{z}_d = \mathbf{y}\mathbf{b}. \tag{47}$$

The duality theorem of linear programming states that $\min \mathbf{z}_p = \max \mathbf{z}_d$ when the primal and dual systems are both feasible. Thus, we must find a solution such that

$$\mathbf{y}\mathbf{b} = \mathbf{c}\mathbf{x}. \tag{48}$$

We can convert the LP inequalities to equalities by introducing slack variables $\mathbf{v}$ and $\mathbf{u}$, subject to nonnegativity constraints

$$\mathbf{A}\mathbf{x} - \mathbf{v} = \mathbf{b}, \qquad \mathbf{v} \geq \mathbf{0}, \qquad \mathbf{x} \geq \mathbf{0}, \tag{49}$$

$$\mathbf{A}^T\mathbf{y} + \mathbf{u} = \mathbf{c}, \qquad \mathbf{u} \geq \mathbf{0}, \qquad \mathbf{y} \geq \mathbf{0}. \tag{50}$$

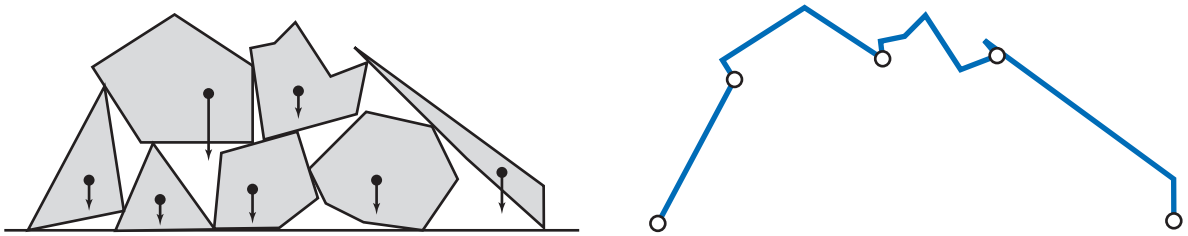
Then, we can convert the LP to the LCP

$$\mathbf{B}\mathbf{z} + \mathbf{q} = \mathbf{w}, \qquad \mathbf{z} \geq \mathbf{0}, \qquad \mathbf{w} \geq \mathbf{0}, \qquad \mathbf{z} \cdot \mathbf{w} = 0 \tag{51}$$

where

$$\mathbf{B} = \begin{bmatrix} \mathbf{0} & -\mathbf{A}^T \\ \mathbf{A} & \mathbf{0} \end{bmatrix}, \qquad \mathbf{z} = \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix}, \qquad \mathbf{q} = \begin{bmatrix} \mathbf{c} \\ -\mathbf{b} \end{bmatrix}, \qquad \mathbf{w} = \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \end{bmatrix}. \tag{52}$$

We believe that we can solve this LCP with an algorithm that reuses computation, in order to efficiently solve the LP. Dantzig's algorithm for LCPs, which we have shown to work on our dynamics formulation, is not likely to work on this formulation in the general case, as it relies on $\mathbf{B}$ being positive definite or positive semidefinite. Finding a solution to the general case is left as an open problem.



(a) An assembly with complete geometry known.

(b) An assembly with only surface geometry known.

Figure 24: We can answer questions about assembly stability without complete knowledge of its geometry.

4.5.2 Questions about strong stability

Recall that to determine strong stability of a structure, we must show that no solutions exist to the system dynamics that indicate the structure is in motion. Known solutions to this problem take exponential time [4, 25]. We propose an approach to solving for strong stability that, if viable could be more tractable.

Consider the interpretation of the system dynamics given by the LCP in equations 1,2, and 3, where $\mathbf{f}$ represents the force magnitudes at contact points, and $\mathbf{a}$ represents contact accelerations. Is it possible to rephrase the LCP such that a basic feasible solution is required to contain some a_i in the set of basic variables? If so, for nondegenerate matrices $\mathbf{A}$, the existence of a solution would indicate that a structure is not strongly stable. If no solution exists, then the structure is strongly stable.

5 Stability of assemblies

In previous sections, we have considered the problem of determining forces between objects in an assembly; in this section we ask how to find the set of external forces an assembly can withstand. Our goal is to take steps toward characterizing precisely what we can learn about the stability of a structure by pressing on it. For many real-world applications this is a more important question, since it is common that we know very little about the geometry of a structure. For example, obtaining precise geometric information about the interior a rubble pile in an urban search-and-rescue environment, if even possible, would be expensive, time-consuming, and potentially dangerous. In this section, we address the multibody stability problem assuming only geometric knowledge of the *surface* of an assembly.

Imagine, for example, how a person might go about crossing a small creek with rocks visible not far below the surface. If the person could see a path of rocks, she might put a foot in to test the stability of the first rock, and if it seemed to be stable, she might stand on this rock and test the next rock, and so on until she reached the other side. If some rock were deemed too unstable, this person might choose not to stand on it, but to try a different rock instead. If she were particularly clever, she might even use a stick to poke at some of the rocks, to decide whether they could bear her weight before wetting her feet.

How can we quantify what humans seem to know innately – that if I push on this surface in this way and it doesn't budge, the odds are good that it will hold my weight? Such information might prove useful in answering the following types of questions:

1. Consider a rubble pile in an urban search-and-rescue environment. We would like to send a 200-pound person across the pile, but we do not know if it will be safe for him. We have a fleet of ten 20-pound robots we can use to obtain information about the stability of the pile. What is the most effective strategy for deploying them to determine a safe path for the human?

2. A 200-pound person has walked across the rubble pile, marking each place he stepped. Can we now send a 500-pound robot across the pile without causing it to collapse? Can we send two 100-pound people? What paths should each of them follow?

3. A rover on Mars reaches a patch of rocky, unstable terrain that it must cross. The robot risks slipping, getting wedged between rocks and being unable to free itself; loosening a rock that might fall and crush it; or collapsing a region of the terrain and falling and breaking. If the rover is equipped with a stick it can use to apply a force that varies depending on the angle of the press, how can it use this tool to find a safe path across the terrain?

4. We are assigned the task of developing a fixture that will stabilize an engine against part insertion forces during assembly. Specifications are unavailable, but we have access to an assembly line of robots that can put together parts in precisely the same way each time. How can we develop the required fixture with only the resources we have?

5. A part of the engine has gotten jammed during insertion. How should we instruct a robot to press on it to unjam it?

Our goal in this section is to approach answering questions like these by identifying stability characteristics that can be obtained experimentally. We would additionally like to take steps toward developing methods for using these characteristics to effectively and efficiently partition a surface into stable and unstable regions. One method is as follows. Given knowledge of an assembly's surface (which we assume can be easily obtained), describe the set of forces we can apply to the surface as a shape in wrench space. We call this shape the *space of applicable wrenches*. Select points on this shape, such that the convex hull of the points contains the shape. Test each point by physically pressing on the surface. We call the convex hull of the stable points the *known stability cone*, and every point inside it is a wrench we can stably apply to the surface. We can map these regions back to the surface to identify the set of stable presses.

If the known stability cone contains the space of applicable wrenches, then we are done: every press we can apply to the surface is stable. How we handle unstable points depends on the specific problem we are addressing, and our model of it. In questions 1 and 2 above, we might require that if we push on the structure and it collapses, we begin the procedure again from the beginning. We might relax this constraint by assuming that if, while pressing, we sense that only one block along the top has moved, we can simply replace it and continue. Alternatively, in questions like 4 and 5 above, we assume that we can reset the structure after an unstable press. This assumption allows us to learn more about stable regions of a structure, since we can use this information to find the boundaries of the stability cone.

In the described approach, we might attempt to maximize the volume of the known stability cone in wrench space. But such an exhaustive approach might be unnecessary. If we can model the wrenches we will apply to the structure during a given task, we might instead try to contain only this set of wrenches in the known stability cone.

In the general case, the goal of the assembly stability problem is to partition wrench space (and, as a result, surface space) into stable and unstable regions. We consider several simplified models, which we describe along three axes: surface shape, information, and press type.

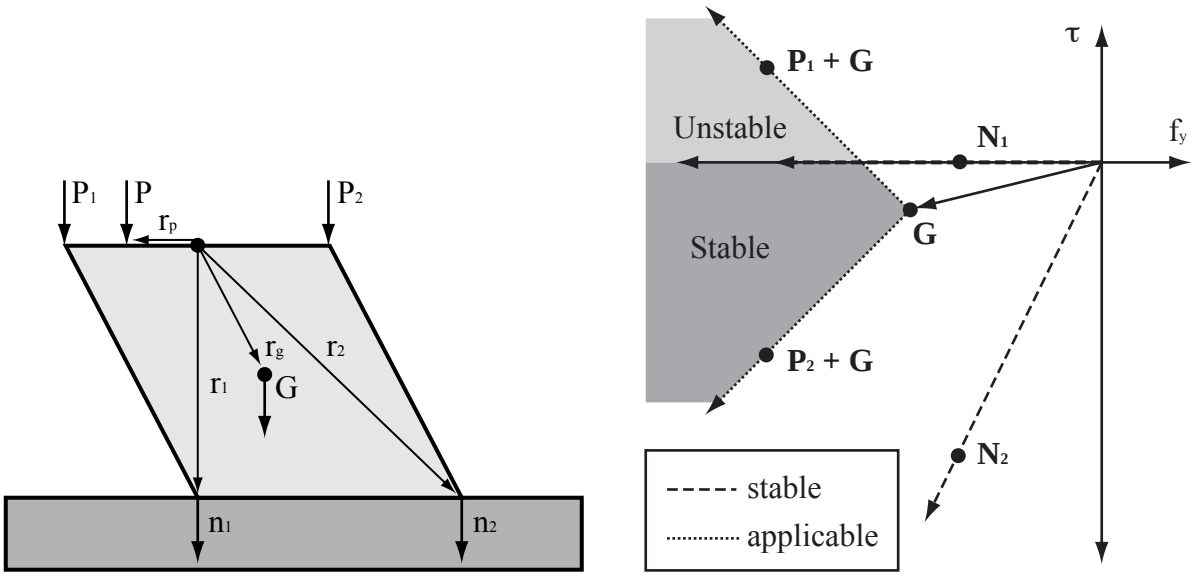
In section 5.1, we present the mathematical tools we use to address this problem, and in section 5.2 we describe the space of models we consider. In subsequent sections, we pose the following simplified question for different models: "Given k presses on a structure about which we have limited knowledge of geometry, what is the space of wrenches that we can apply without causing the structure to collapse?" In section 5.3 we give an expression for the magnitude of a stable press between two stable presses along a polygonal edge. In section 5.5 we discuss how the approach described above can be applied to curved surfaces as well. However, there is a caveat: we show that there exist surfaces for which our approach does not work, since no set of frictionless normal presses tells anything about presses at other points on the surface.

5.1 Background

In this section, we introduce the mathematical framework we use in analyzing the assembly stability problem.

Consider figure 24(b). We define a *press* as a pair $P = (\mathbf{f}, \mathbf{l})$ where $\mathbf{f}$ is a known force applied at location $\mathbf{l}$ on the assembly surface. A press applies a wrench

$$\mathbf{w} = \begin{bmatrix} \mathbf{f} \\ \tau \end{bmatrix} \tag{53}$$



(a) Simple one-block assembly with polygonal edges.

(b) Wrench space of one-block assembly with known geometry.

Figure 25: If we know the geometry of a structure, we can determine the space of stable and unstable presses.

to a single object on the surface, where $\mathbf{w}$ is a three-dimensional vector for a planar assembly, and a six-dimensional vector for a three-dimensional assembly. A press is said to be unstable if any non-empty subset of objects in the assembly shifts when the press is applied, and stable if all objects remain fixed. If we press in several places along the surface of the assembly, we can use what we learn about the stability of each press to describe the space of wrenches the structure can stably withstand. In terms of computation reuse, we say that we *experimentally* obtain a set of solutions that we can then reuse to make guarantees about the stability of the assembly.

Consider the simple example in figure 25(a), where $\mathbf{n}_1$ and $\mathbf{n}_2$ are the outward-pointing contact normals, and P_1 and P_2 are presses at the extreme ends of the top edge of the rhombus. If we know the geometry of this structure, we can use the force balance equations

$$\mathbf{J}^T\mathbf{f} \quad = \quad \mathbf{F}^{\text{ext}}, \mathbf{f} \geq \mathbf{0} \tag{54}$$

to determine the space of stable presses, where $\mathbf{J}^T$ is the transpose of the system Jacobian, $\mathbf{f}$ is the vector of unknown contact force magnitudes, and $\mathbf{F}^{\text{ext}}$ is the vector of external wrenches, as discussed in section 3.1.1. The external wrench vector $\mathbf{F}^{\text{ext}}$ can be broken down into the external wrench due to gravity, and the external wrench due to applied presses,

$$\begin{bmatrix} \mathbf{n}_1 & \mathbf{n}_2 \\ \mathbf{r}_1 \times \mathbf{n}_1 & \mathbf{r}_2 \times \mathbf{n}_2 \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} \quad = \quad \begin{bmatrix} P \\ P \times \mathbf{r}_p \end{bmatrix} + \begin{bmatrix} G \\ G \times \mathbf{r}_g \end{bmatrix}, f_1, f_2 \geq 0. \tag{55}$$

Any press P for which this linear program has a solution is weakly stable (weak and strong stability are discussed in greater detail in section 2.1).

For the remainder of this section, we will consider any structure for which a solution exists to equation 55 to be stable. Although is is possible that this assumption is not physically accurate, since it does not prove that the structure is strongly stable, it is a common convention in assembly planning literature. The difficulty with determining strong stability with Coulomb friction is that known methods for doing so take exponential time. However, it is worth recalling that Coulomb friction is an experimentally determined model. In those cases where we have experimentally verified our results, our predictions match them very well.

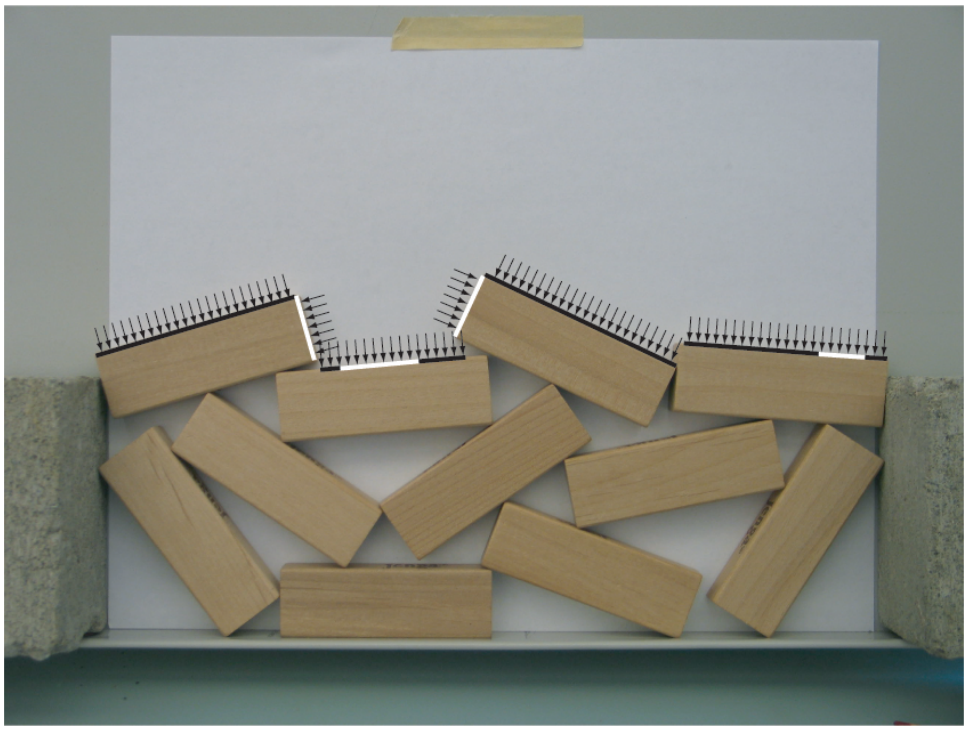


Figure 26: Jenga pile with stability profile drawn in. Black regions are unstable. White regions support presses of nonzero magnitude. Stability profile was obtained by sampling the surface at 0.3 cm intervals.

Graphically, equation 55 can be represented as the wrench space in figure 25(b). Let $\mathbf{N}_1$ and $\mathbf{N}_2$ be the wrenches applied by unit contact forces $\mathbf{n}_1$ and $\mathbf{n}_2$, and let $\mathbf{P}_1 + \mathbf{G}$ and $\mathbf{P}_2 + \mathbf{G}$ be the wrenches applied by presses P_1 and P_2 with gravity present. We call the positive linear span of $\mathbf{N}_1$ and $\mathbf{N}_2$ the *stability cone*; all wrenches that lie in this cone are stable. The positive linear span of $\mathbf{P}_1 + \mathbf{G}$ and $\mathbf{P}_2 + \mathbf{G}$ is called the *space of applicable wrenches*; all the wrenches we can apply by pressing the top edge of the rhombus lie this this space. The intersection of these two spaces gives us the set of stable wrenches that can be applied to this edge. The set of wrenches we can apply that lie outside the stability cone are unstable. This partitioning of the assembly surface into stable and unstable regions is called the assembly's *stability profile*, and can be mapped back to the surface of the assembly. Figure 26 gives an example of a complex structure for which the stability profile was obtained by sampling the surface at regular intervals. Methods regarding the collection of experimental data are discussed in greater detail in section 5.4.

If we do not know the geometry of the structure but know where some stable presses may be applied along the surface, we can still determine sections of an assembly's stability profile. It is straightforward to show that an object can resist any positive linear combination of known stable wrenches. The proof of this is given in [60], and the theorem is stated as follows.

Theorem 3 (Theorem 6.1 of Romney) *If an assembly is stable against an external wrench* $\mathbf{W}_0$*, and it is also stable against an external wrench* $\mathbf{W}_1$*, then it is stable against any external wrench* $u\mathbf{W}_0 + v\mathbf{W}_1$*, for* $u, v \geq 0$.

By this theorem, if we know wrenches $\mathbf{W}_1$, $\mathbf{W}_2$, and $\mathbf{W}_3$ are stable, we can determine that $\mathbf{W}_4$ is also stable if a solution exists to the linear program

$$\mathbf{W}\mathbf{x} = \mathbf{W}_4, \mathbf{x} \geq \mathbf{0}, \tag{56}$$

where

$$\mathbf{W} = \begin{bmatrix} \mathbf{W}_1 & \mathbf{W}_2 & \mathbf{W}_3 \end{bmatrix}. \tag{57}$$

Consider the geometric implication of this theorem. A wrench applied to an assembly represents a single point in wrench space. For any stable press, a ray from the origin through the corresponding point is a *stable ray*. The convex hull of these rays gives us the *known stability cone*. It is important to distinguish between the known stability cone, of the type shown in figure 28, and the actual stability cone, of the type shown in figure 25(b). Given some set of stable presses, we can determine the known stability cone, but without additional information, we cannot partition the wrench space outside this cone into stable and unstable regions. In contrast, if we assume complete geometric information, as we do in the case of the actual stability cone, we can partition the space completely.

5.2 Problem space

In this section, we map out the problem space in terms of the parameters that vary for each case of the problem we consider. In all cases, we assume contacts forces are unilateral, and that the geometry of the visible surface can be obtained.

The first variable is *surface shape*: we consider polygonal models in section 5.3 and continuous parametric models in section 5.5. We do not consider three-dimensional surfaces, but conjecture that our methods are similarly applicable.

The second variable we consider is *information*: how much information do we have about the geometry and inertial properties of the structure? If we have complete information, we can determine stability using a physical model, as discussed in the previous section. In the following sections we consider first the case where we know the center of mass of objects at the surface of an assembly, and therefore the gravitational wrench on these objects; and second, the case that we have no information about the structure other than the geometry of its surface. If we know the mass characteristics of surface objects, we consider both the case that $\mathbf{G} = \mathbf{0}$ and $\mathbf{G} \neq \mathbf{0}$.

Finally, we discuss varying *press type*: either pressing with a frictionless finger normal to the surface (a frictionless normal press), or with a frictional finger that can apply forces within the friction cone at a point on the surface.

5.3 Polygonal surface model

For a polygonal edge with frictionless normal presses, we can state the problem as:

A given planar assembly is stable under gravity. It remains stable under two separate presses on the same polygonal edge: A, of magnitude a, and B, of magnitude b and distance x_b from A, as in figure 27(a). Find the magnitude c of the maximum guaranteeably-stable press C between A and B as a function of x_c, the distance of C from A.

Choose coordinates so that the origin is at the contact point of press A, and the x-axis is coincident with the polygonal edge, as in figure 27(b). Then, the wrenches applied by presses A, B and C are

$$\mathbf{A} = \begin{bmatrix} 0 \\ a \\ 0 \end{bmatrix}, \qquad \mathbf{B} = \begin{bmatrix} 0 \\ b \\ bx_b \end{bmatrix}, \qquad \mathbf{C} = \begin{bmatrix} 0 \\ c \\ cx_c \end{bmatrix} \tag{58}$$

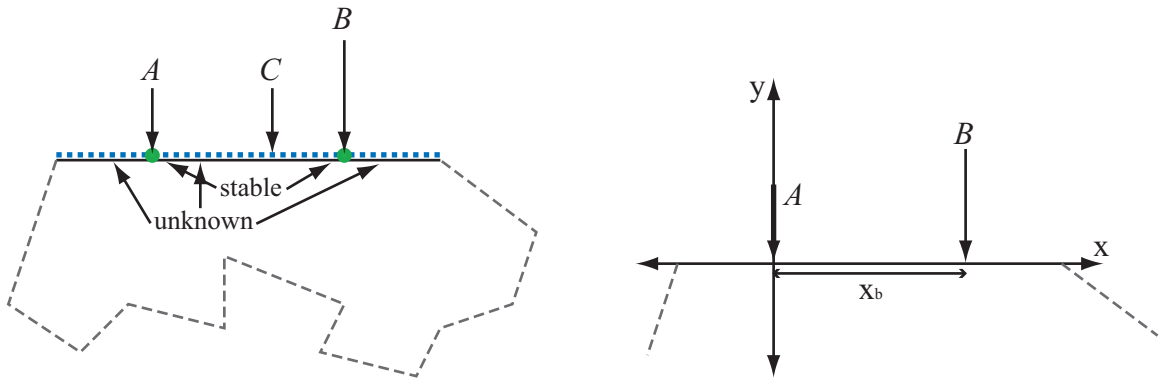
where c is unknown.

Since the f_x-coordinate of each press is zero, we can draw wrenches $\mathbf{A}$ and $\mathbf{B}$ as points in the $f_y\tau$-plane of wrench space, as in figures 28, 29 and 30. If there is gravity, then there is a third stable wrench. In section 5.3.1, we consider first the case that $\mathbf{G}$ is zero, then the case that $\mathbf{G}$ is nonzero but known. Finally in section 5.3.2, we consider the case that we know nothing about $\mathbf{G}$.

5.3.1 Gravity is known

Case 1: $\mathbf{G} = \mathbf{0}$. If gravity is absent then the known stability cone is exactly the space of applicable wrenches between A and B, as shown in figure 28. A press of any magnitude between two stable presses is also stable. The following is a corollary to theorem 3.

Corollary 1 *If $\mathbf{G} = \mathbf{0}$, C is stable for all $x_c, 0 \leq x_c \leq x_b$.*



(a) Edge with stable presses A and B and unknown press C. By convention, dashed lines represent unknown geometry

(b) Coordinate selection in surface space, with origin at contact point of A.

Figure 27: Illustration and coordinate selection for polygonal-edge problem.

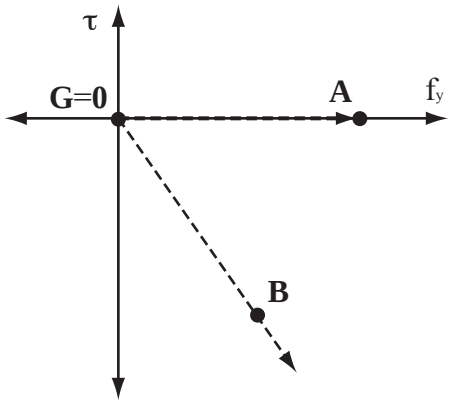


Figure 28: Known stability cone in wrench space, for $\mathbf{G} = \mathbf{0}$.

Proof: By theorem 3, any wrench that lies in the positive linear span of wrenches $\mathbf{A}$ and $\mathbf{B}$ is stable. The slopes of rays $\mathbf{O} + t\mathbf{A}$, $\mathbf{O} + u\mathbf{B}$, and $\mathbf{O} + v\mathbf{C}$, are 0, x_b, and x_c, respectively. Thus, if $0 \leq x_c \leq x_b$, $\mathbf{C}$ lies in the positive linear span of $\mathbf{A}$ and $\mathbf{B}$, so press C is stable. ■

Case 2: $\mathbf{G} \neq \mathbf{0}$. If gravity is non-zero, the known stability cone no longer strictly contains the space of applicable wrenches, as is illustrated in figure 29. However, since we know $\mathbf{G}$, we know the edge is stable under three applied wrenches: $\mathbf{G}$, $\mathbf{A} + \mathbf{G}$, and $\mathbf{B} + \mathbf{G}$. We can draw the known stability cone with edge rays $\mathbf{O} + u(\mathbf{G})$, $\mathbf{O} + v(\mathbf{A} + \mathbf{G})$ and $\mathbf{O} + w(\mathbf{B} + \mathbf{G})$. By theorem 3, any wrench that falls in this cone is stable. If in addition we know a magnitude for which a press at $\mathbf{A}$ or $\mathbf{B}$ is unstable, we can make guarantees regarding the space of unstable presses as well, as discussed in the following section.

5.3.2 Gravity is unknown

In the following section, we examine how much we can guarantee about the stability of an assembly when we know nothing about the mass characteristics of objects along its surface. Although we do not explicitly do so, we effectively identify the space of wrenches that are stable under all possible directions of gravity. This space turns out to be a

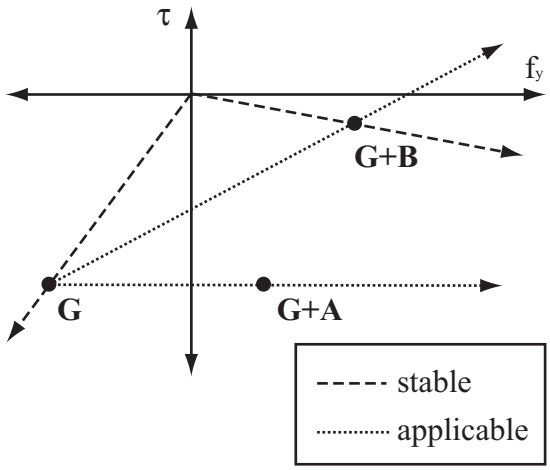


Figure 29: If $\mathbf{G} \neq \mathbf{0}$, the space of applicable forces may not be contained by the known stability cone.

triangle in wrench space. We can use this fact to determine the magnitude of presses that must be stable for any assembly for which we have some set of known stable presses. Given additional information about unstable presses, we may make further guarantees about the assembly's stability profile.

Let $\mathbf{A}' = \mathbf{G} + \mathbf{A}$ and $\mathbf{B}' = \mathbf{G} + \mathbf{B}$. Let $\triangle\mathbf{G}\mathbf{A}'\mathbf{B}'$ be the convex hull of points the $\mathbf{G}$, $\mathbf{A}'$ and $\mathbf{B}'$.

Lemma 1 *Every wrench in $\triangle\mathbf{G}\mathbf{A}'\mathbf{B}'$ is stable.*

Proof: By theorem 3, any positive linear combination of $\mathbf{G}$, $\mathbf{A}'$ and $\mathbf{B}'$ is stable. $\triangle\mathbf{G}\mathbf{A}'\mathbf{B}'$ always lies in the positive linear span of these points. ■

Given that $\mathbf{G} + \mathbf{A}$ and $\mathbf{G} + \mathbf{B}$ are known stable wrenches, the smallest space of stable wrenches occurs when $\mathbf{G} + \mathbf{A}$ and $\mathbf{G} + \mathbf{B}$ lie on the same edge of the stability cone, and every point on the other side of this edge is unstable. The following theorem gives the equation for points that lie on the line between $\mathbf{G} + \mathbf{A}$ and $\mathbf{G} + \mathbf{B}$. Presses of this magnitude must be stable for all assemblies.

To simplify the following equations, we choose wrench space coordinates with the origin at $\mathbf{G}$.

Theorem 4 *If A and B are stable presses of magnitudes a and b, respectively, and B is a distance x_b from A, then a press C, a distance x_c from A, with magnitude*

$$c = \frac{abx_b}{bx_b - x_c(b-a)}, \tag{59}$$

is stable.

Proof: By lemma 1, all wrenches that lie in $\triangle\mathbf{G}\mathbf{A}\mathbf{B}$ are stable, regardless of the value of $\mathbf{G}$. Let $\mathbf{E}$ be the vector with a unit projection onto the f_y-axis and slope x_c, as shown in figure 30. The value of c for which the ray $\mathbf{O} + c\mathbf{E}$ intersects the line segment $\overline{\mathbf{AB}}$ gives us magnitude c of a press C that must be stable.

We can find this intersection as follows. The equation for line segment $\overline{\mathbf{AB}}$ is given by $\mathbf{A} + s(\mathbf{B} - \mathbf{A}), s \in [0, 1]$. This intersects ray $\mathbf{O} + c\mathbf{E}$ where

$$\mathbf{A} + s(\mathbf{B} - \mathbf{A}) = \mathbf{O} + c\mathbf{E} \tag{60}$$

$$\begin{bmatrix} a \\ 0 \end{bmatrix} + s \begin{bmatrix} b-a \\ bx_b \end{bmatrix} = c \begin{bmatrix} 1 \\ x_c \end{bmatrix} \tag{61}$$

for some values of s, c. Breaking these vectors into f_y and τ coordinates, we have two variables and two equations, and can solve for c at the point of intersection.

$$a + s(b-a) = c \tag{62}$$

$$sbx_b = cx_c \tag{63}$$

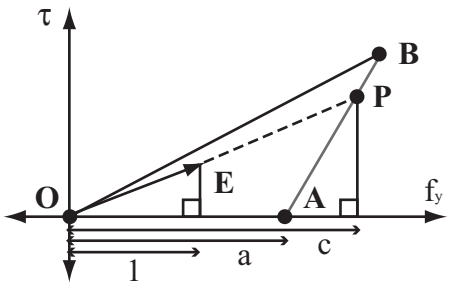


Figure 30: Wrench space construction for theorem 4. The f_y-projection of the intersection point gives us c.

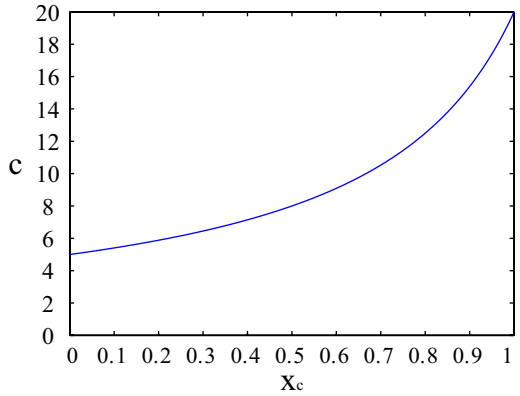


Figure 31: Maximum magnitude of press C as a function of its distance from A.

Solving equation 62 for s,

$$s = \frac{c-a}{b-a}, \tag{64}$$

and then substituting the value of s into equation 63, we find c:

$$\frac{c - abx_b}{b-a} = cx_c \tag{65}$$

$$c\frac{bx_b - x_c(b-a)}{b-a} = \frac{abx_b}{b-a} \tag{66}$$

$$c = \frac{abx_b}{bx_b - x_c(b-a)}. \tag{67}$$

■

A graph of this function for $a = 5, b = 20$ and $x_b = 1$ is shown in figure 31. An alternate proof of theorem 4 is given in Appendix A.

If $a = b$, the term $(b-a) = 0$, so the independent variable x_c is removed from the equation, and we have $c = a$. Thus, as a consequence of theorem 4, given two stable presses of the same magnitude along a polygonal edge, we can predict that a press with the same magnitude will be stable at any point between them.

Theorem 4 allows us to guarantee that some press between two stable presses will also be stable. Moreover, it allows us to characterize the stable magnitude of this press, either conservatively or precisely, as discussed in greater detail in section 5.4.

An immediate consequence of this theorem is that, given any two stable presses along a polygonal edge, at all locations between them there is some stable press. This allows us to make predictions about unstable regions of the assembly's surface as well.

Define a function $stable(P)$, which takes a press $P = (\mathbf{f}, \mathbf{l})$.

$$stable(P) = \begin{cases} 1 & \text{if } P \text{ is stable for } ||P|| > 0 \\ 0 & \text{otherwise.} \end{cases}$$

Corollary 2 *The preimage of $stable(P)$ is an interval.*

Proof: Assume there is some unstable press U that lies in the domain of $stable(P)$ between two stable presses X and Y. U must be a linear combination of some X and Y. Then, by theorem 3, U is stable. ■

If, in addition to a stable press, we know some press on an edge of the assembly is unstable, then we can designate a region of wrench space as unstable.

Corollary 3 *If $\mathbf{B}$ is a stable wrench and $(1+\epsilon)\mathbf{B}$ is unstable for $\epsilon > 0$, then $\forall \delta > \epsilon$, $\mathbf{O} + t[(1+\delta)\mathbf{B}]$ is a ray of unstable wrenches.*

Proof: By corollary 2, the space of stable presses is contiguous. ■

The implications of corollary 3 are useful in reducing the number of points we must test along a polygonal edge. In surface space, any press with magnitude greater than the known unstable press is also unstable. Additionally, if the known unstable press is some distance $x_u > 0$ away from a stable press, then all presses at distance $x \geq x_u$ must also be unstable. Thus, we can employ a binary search approach to finding the boundaries of the stable region once we have obtained both a stable and an unstable press. This is discussed in greater detail in section 5.6.2.

5.4 Experimental results

In this section, we describe two experiments conducted to determine the correctness of our theoretical results. In the first, we were concerned with showing that the magnitude function given by theorem 4 is correct when the geometry of the structure is unknown. We also showed that the magnitude function is conservative, as expected. In the second, we wished to show that there exist structures for which the magnitude function gives a precise description of the maximum press magnitude that can be applied between two known stable presses. As might be expected, more computational effort went into devising a structure for which this is true. In the following sections, we describe the objectives, materials, procedures and results of both these experiments, and discuss our findings.

5.4.1 Experiment 1: does magnitude function predict correctly?

Objective. We wish to determine whether, for a structure with unknown geometry, the magnitude function given by theorem 4 is correct. The only requirement for this structure is that it be asymmetric, so we can easily choose A and B of different magnitudes. Masses and frictional characteristics of objects in the assembly should not be obtained. Dimensions of objects are only obtained so that the structure can be precisely reassembled when it is disturbed.

Materials. The materials used in this experiment are:

- Wooden blocks with polyhedral edges
- Concrete bricks
- Template for constructing assembly, given in figure 44
- Flat backboard to which template can be affixed
- Force gauge, McMaster-Carr part #2115T14

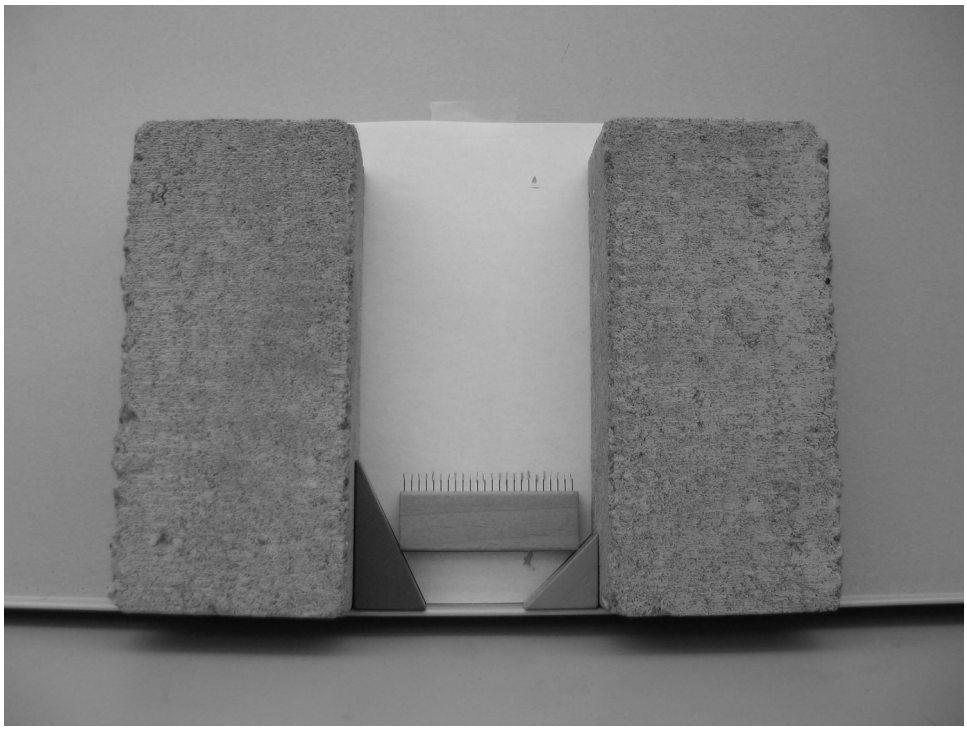


Figure 32: Assembly used for experiment 1.

Procedure. Affix the template to the backboard, and assemble the structure in the configuration specified by the template. Place concrete bricks on either side of the structure. The complete assembly is shown in figure 32. The force gauge is a linear tension and compression, spring-operated scale. It has gradations of 0.1 kg and accuracy of +/- one gradation. Use the force gauge to determine the maximum magnitude press that can can be applied at each of the 0.3 cm intervals marked on the template. It is important to apply each press at a perpendicular to the surface, and to increase the force slowly, so that the magnitude can be read prior to the structure giving way. After each force magnitude has been obtained, reassemble the structure, and continue sampling points until all test points have been collected.

Results and discussion. The results of this experiment are displayed in figure 33, with the magnitude curves predicted by theorem 4 drawn in between all pairs. Please note that the straight lines drawn in gray between data points are simply to aid visualization of the curve, and have no physical significance.

These results support the predictions of theorem 4. For any two stable presses, an intermediate press either lies on or above the predicted curve.

We might ask what the best order to select presses is. We leave a formal answer to this question as an open problem, and discuss it in greater detail in section 5.6.2. One approach is to select presses to maximize the area under the stability curve.

There are a number of possible error sources in this experiment. The first is that, since we test the assembly to failure for each press, the exact geometry of the structure may change slightly after each point is sampled. We work against this as best we can by providing a template to help align the blocks precisely. A second source is in our force gauge readings. We increase the force slowly so that we can read force measurements prior to failure, but it is difficult to be certain of the exact value at the moment of failure, since the structure is falling apart. Finally, our theoretical results apply to planar structures with frictionless presses. Although we approximate this as best we can with flat blocks and normal presses, there is room for error to creep in here as well, since this is in fact a three-dimensional structure and our force gauge is not actually frictionless.

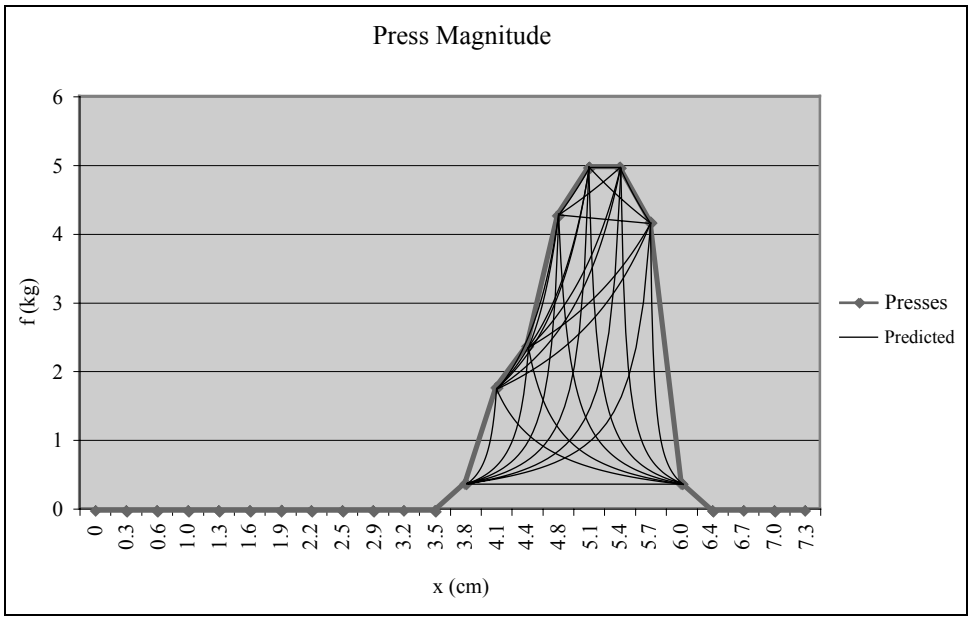


Figure 33: Results of experiment 1. For any two stable presses, the predicted curve is exceeded.

5.4.2 Experiment 2: are there instances where magnitude function predicts precisely?

Objective. In this experiment, we wish to show that there exist structures for which the magnitude function given in theorem 4 is precise. We do this by first calculating the geometry of an assembly for which both stable pushes lie on the edge of the stability cone. We then build this structure and determine whether our experimental results match our predictions.

Materials. The materials used in this experiment are:

- One clay brick
- Two triangular wooden blocks
- Force gauge, McMaster-Carr part #2115T14
- Ruler
- Sharpie permanent marker

Structure design. In this subsection, we discuss how the geometry is calculated for the assembly. The equations used here are discussed in greater detail in sections 3.1.1 and 5.1.

We use a wrench space construction to derive the geometry of the experimental assembly. Let press A be a frictionless normal press at the reference point on the polygonal edge, and let its magnitude a be the maximum magnitude for which a press at this point is stable. Let B be a similarly defined press at a distance x_b from the reference point. We would like to select $\mathbf{A}$ and $\mathbf{B}$ so that $\mathbf{G}+\mathbf{A}$ and $\mathbf{G}+\mathbf{B}$ lie on the same edge of the stability cone. This way, the line between them gives not only the maximum press magnitude we can guarantee to be stable, but also the maximum press magnitude that is actually stable for this structure. Figure 34 gives an instance of this construction in the $f_y\tau$-plane of wrench space, with the opposite edge of the stability cone going through $\mathbf{G}$.

The edges of the stability cone give us the columns of $\mathbf{J}^T$ as discussed in section 5.1, and $\mathbf{J}^T$ also describes the location of contact constraints in an assembly. Thus, we can derive the desired structure's geometry by finding the Jacobian from the stability cone we've drawn.

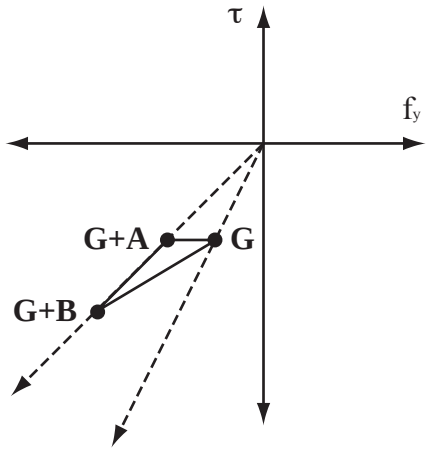


Figure 34: Wrench space construction for experiment 2.

We require that $\mathbf{G}+\mathbf{A}$ and $\mathbf{G}+\mathbf{B}$ lie on a line through the origin, as expressed by

$$\alpha(\mathbf{G}+\mathbf{B}) = \mathbf{G}+\mathbf{A}, \qquad \alpha > 0, \tag{68}$$

and that both $\mathbf{G}+\mathbf{A}$ and $\mathbf{G}+\mathbf{B}$ lie on one edge of the stability cone, which is true if both equations

$$\mathbf{J}^T \begin{bmatrix} f_1 \\ 0 \end{bmatrix} = \mathbf{G}+\mathbf{A}, f_1 \geq 0, \qquad \mathbf{J}^T \begin{bmatrix} f_1 \\ 0 \end{bmatrix} = \mathbf{G}+\mathbf{B}, f_1 \geq 0 \tag{69}$$

have a solution. Finally, so that $\mathbf{G}$ lies on the other edge of the stability cone, the equation

$$\mathbf{J}^T \begin{bmatrix} 0 \\ f_2 \end{bmatrix} = \mathbf{G}, \qquad f_2 \geq 0 \tag{70}$$

must have a solution.

For this experiment, we may select any values for $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{G}$ that satisfy these constraints, and then scale them according to the measured gravitational force. Choosing

$$\mathbf{G}+\mathbf{A} = \begin{bmatrix} -2 \\ -2 \end{bmatrix}, \qquad \mathbf{G}+\mathbf{B} = \begin{bmatrix} -4 \\ -4 \end{bmatrix}, \qquad \text{and} \qquad \mathbf{G} = \begin{bmatrix} -1 \\ -2 \end{bmatrix}, \tag{71}$$

gives wrenches due to presses A and B,

$$\mathbf{A} = \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \qquad \mathbf{B} = \begin{bmatrix} -3 \\ -2 \end{bmatrix}, \tag{72}$$

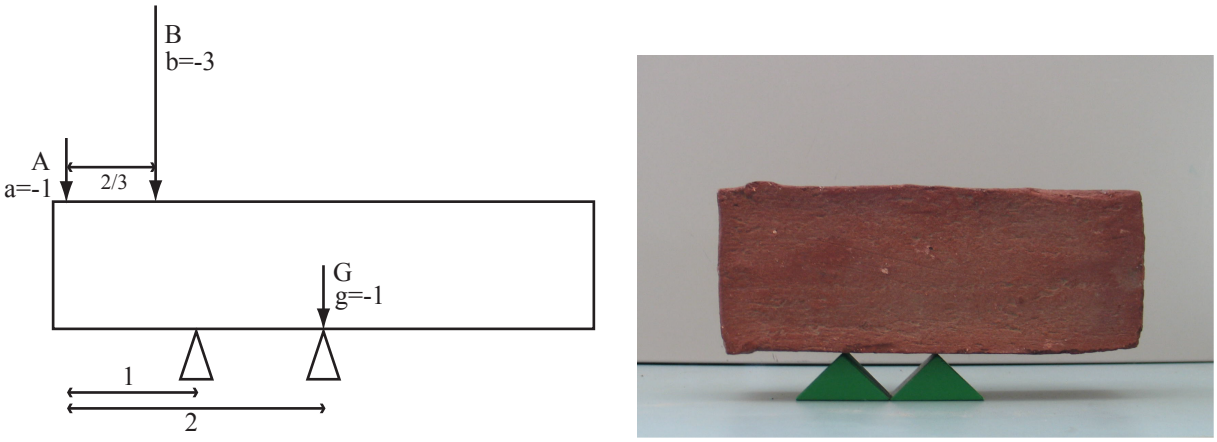
where $a = -1$, $b = -3$ and $x_b = \frac{2}{3}$. A matrix $\mathbf{J}^T$ that satisfies constraints 69 and 70 is

$$\mathbf{J}^T = \begin{bmatrix} 0 & 0 \\ -1 & -1 \\ -1 & -2 \end{bmatrix}. \tag{73}$$

This gives us outward-pointing contact normals

$$\mathbf{c}_1 = \begin{bmatrix} 0 \\ -1 \end{bmatrix}, \qquad \mathbf{c}_2 = \begin{bmatrix} 0 \\ -1 \end{bmatrix} \tag{74}$$

with contact points at locations $x_1 = 1$ and $x_2 = 2$ from the reference point under press A. Figure 35(a) illustrates the ideal geometry derived from this construction. Figure 35(b) shows the physical assembly built from this geometry.



(a) Ideal geometry, as derived from wrench-space construction.

(b) Physical assembly built from ideal geometry.

Figure 35: Physical constructions for experiment 2.

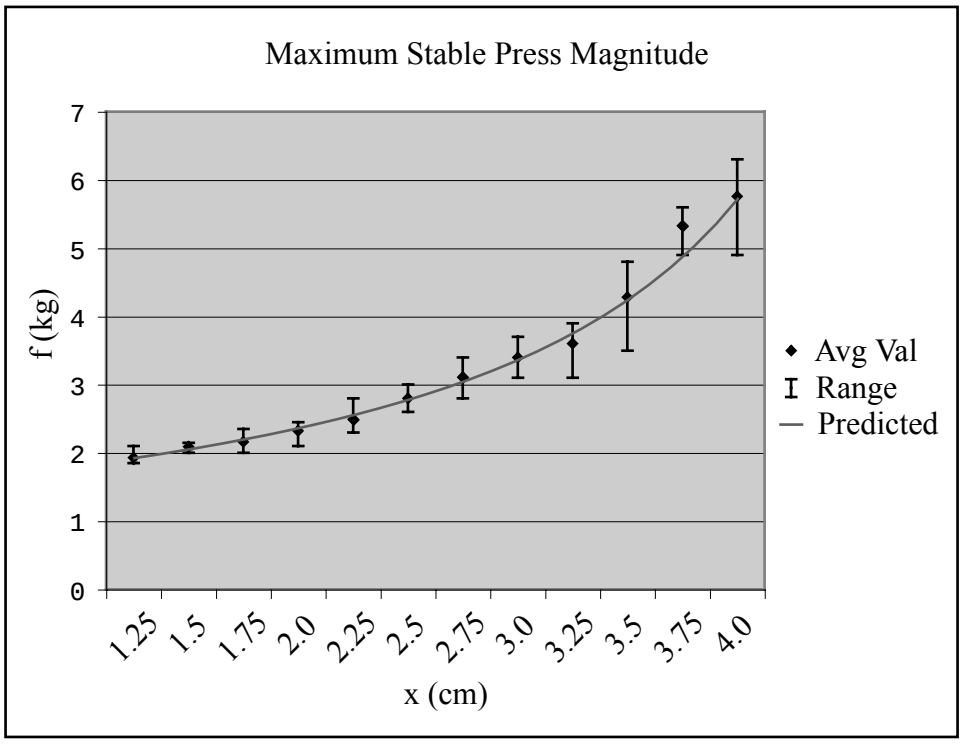


Figure 36: Experiment 2 data with predicted curve superimposed.

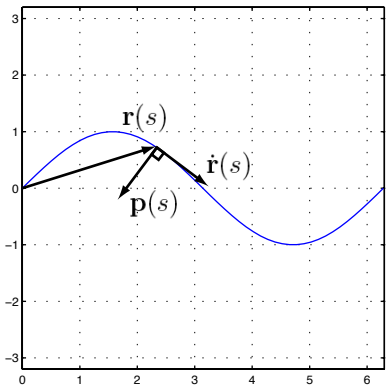


Figure 37: Example of a smooth parametric edge, for $\mathbf{r}(s) = \sin(s)$.

Procedure. Weigh the brick to determine the gravitational force $\mathbf{G}$ on it. Place one of the triangular blocks beneath the center of the brick and move it until the brick is balanced. Mark the center of gravity on the brick with the permanent marker.

Along the top edge of the brick, use the ruler to measure intervals of 0.5 cm, and mark these points with the permanent marker. These will be used to guide the selection of test points, without having to re-measure every time.

Set up the assembly as shown in figures 35(a) and 35(b) by placing one triangular block directly below the center of mass and another one unit to the left of it. Select a point on the far left of the brick and measure the maximum force that can be applied at this point before each press, to ensure consistency between tests. Apply presses at 0.25 cm intervals, reassembling the structure between presses, as described in section 5.4.1.

Results and discussion. The measured weight of the brick was 1.9 kg, and the center of mass was estimated to be 10 cm from the brick's edge. Five sets of force measurements were taken every 0.25 cm along the surface of the block, and averaged together. The closest values to those suggested by our geometric construction were selected as presses A and B. Press A (of magnitude 1.93 kg) was applied at a distance 1.45 cm along the brick, and press B (of magnitude 5.76 kg) was applied at 4.2 cm. The predicted magnitude for C with $a = 1.93$, $b = 5.76$ and $x_b = 2.75$ is plotted is superimposed on top of the measured data in figure 36. Error bars show the range of measured values over the different trials.

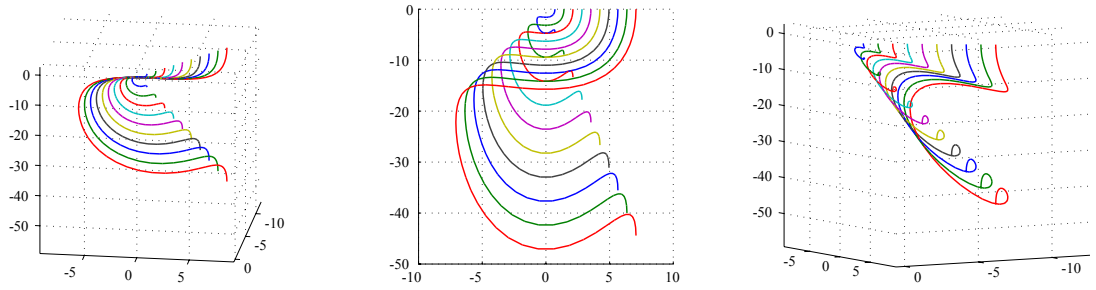
These results confirm that a structure can be constructed for which the magnitude function given by theorem 4 is reasonably precise. The possible sources of error in this experiment are the same as those discussed in section 5.4.1.

5.5 Smooth surface model

In this section, we consider assemblies with surface edges that can be expressed as smooth parametric function $\mathbf{r}(s)$, which are a superset of the straight-line edges we considered in section 5.3. With this change to the underlying model, we address essentially the same question:

A given planar assembly is stable under gravity. It remains stable under k separate presses on an edge of the same object at the surface of the assembly. What is the set of stable presses that can be applied to the structure?

As in the preceding sections, our goal is to obtain experimentally a known stability cone that contains the largest possible set of applicable wrenches. For a smooth parametric curve, however, it is possible that the surface of applicable wrenches is such that a known stability cone defined by a set of k points on the surface contains no points other than these k points. In this section, we show that assemblies exist such that no set of frictionless normal presses can give any information about the stability of other presses.



(a) View of surface. (b) $x\tau$-projection of surface. (c) View of surface.

Figure 38: Space of applicable wrenches for parametric curve $\mathbf{r}(s) = \sin(s)$.

Let $\mathbf{r}(s)$ be the parametric equation of a smooth surface. For a planar surface,

$$\mathbf{r}(s) = \begin{bmatrix} \mathbf{x}(s) \\ \mathbf{y}(s) \end{bmatrix}. \tag{75}$$

The tangent vector to the curve at $\mathbf{r}(s)$ is $\dot{\mathbf{r}}(s)$, and the unit press vector $\mathbf{p}(s)$ is the normalized tangent vector rotated $-90°$,

$$\mathbf{p}(s) = \frac{1}{\sqrt{\dot{\mathbf{x}}(s)^2 + \dot{\mathbf{y}}(s)^2}} \begin{bmatrix} \dot{\mathbf{y}}(s) \\ -\dot{\mathbf{x}}(s) \end{bmatrix}. \tag{76}$$

The torque is

$$\begin{aligned} \tau &= \mathbf{r}(s) \times \mathbf{p}(s) && (77) \\ &= \frac{1}{\sqrt{\dot{\mathbf{x}}(s)^2 + \dot{\mathbf{y}}(s)^2}} (-\mathbf{x}(s)\dot{\mathbf{x}}(s) - \mathbf{y}(s)\dot{\mathbf{y}}(s)). && (78) \end{aligned}$$

Thus, the point $\mathbf{w}(m, s)$ in wrench space due to a frictionless normal press of magnitude m at surface location $\mathbf{r}(s)$ is

$$\mathbf{w}(m, s) = \frac{m}{\sqrt{\dot{\mathbf{x}}(s)^2 + \dot{\mathbf{y}}(s)^2}} \begin{bmatrix} \dot{\mathbf{y}}(s) \\ -\dot{\mathbf{x}}(s) \\ -\mathbf{x}(s)\dot{\mathbf{x}}(s) - \mathbf{y}(s)\dot{\mathbf{y}}(s) \end{bmatrix}. \tag{79}$$

Consider the example in figure 37. The equation of this smooth parametric edge is

$$\mathbf{r}(s) = \begin{bmatrix} s \\ \sin(s) \end{bmatrix}, \tag{80}$$

and the surface of applicable wrenches is described by

$$\mathbf{w}(m, s) = \frac{m}{\sqrt{1 + \cos^2(s)}} \begin{bmatrix} \cos(s) \\ -1 \\ -s - \sin(s)\cos(s) \end{bmatrix}. \tag{81}$$

Several views of this surface in three-dimensional wrench space are shown in figures 38(a), 38(b) and 38(c). In these images, the surface of applicable wrenches is represented by the set of curves corresponding to presses of constant magnitude for $m = 1, 2, \ldots, 10$. We will use this convention for depicting the surface of applicable wrenches for assembly edges elsewhere this section as well.

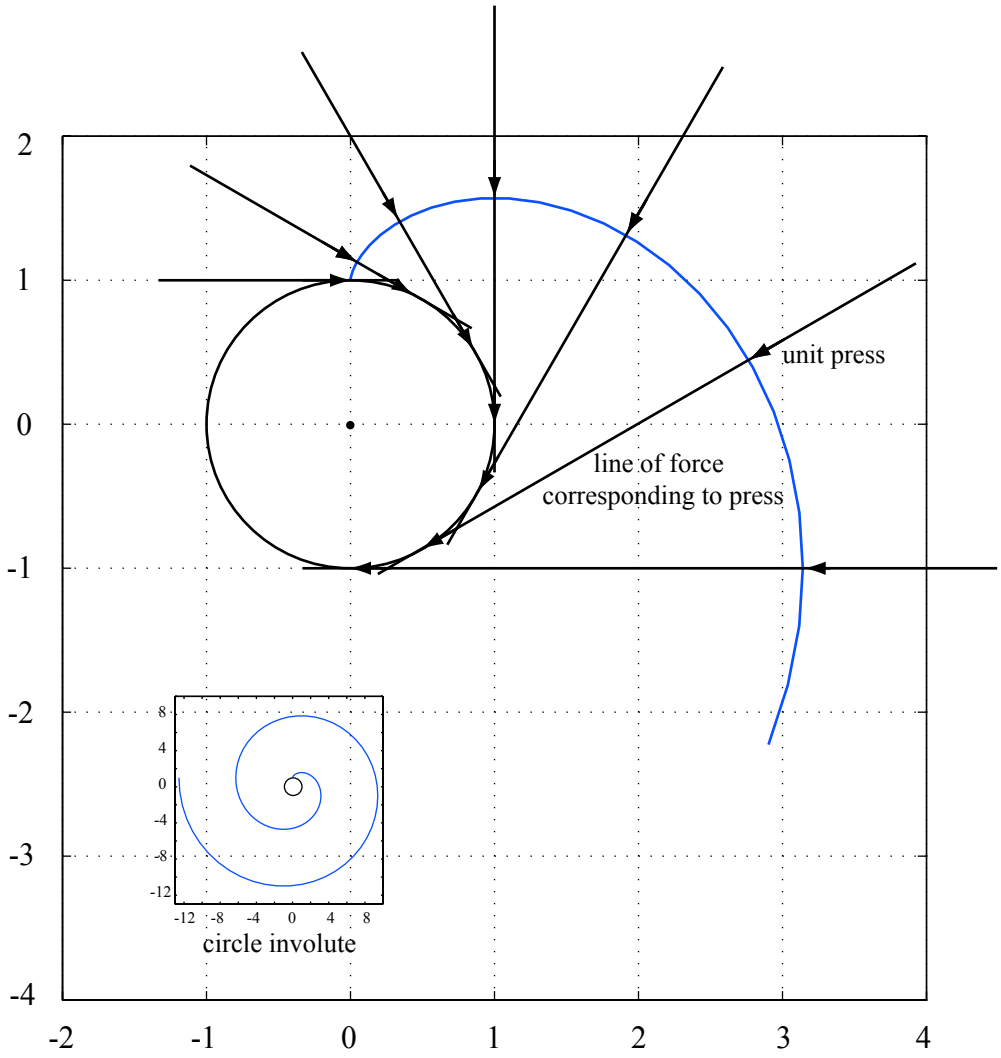


Figure 39: Geometric construction of assembly in figure 40. Every unit press applies a unit torque, thus the surface of applicable presses of constant magnitude lies in a plane.

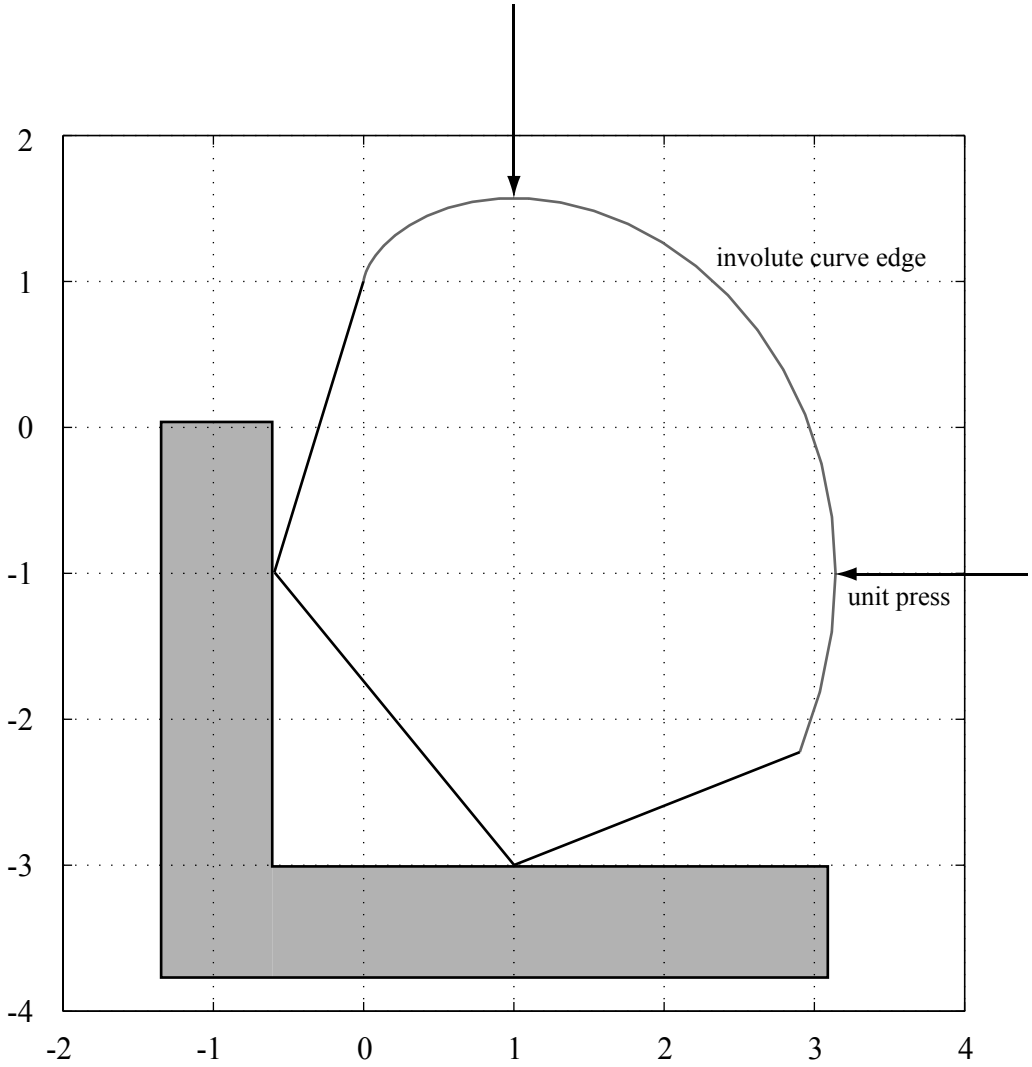
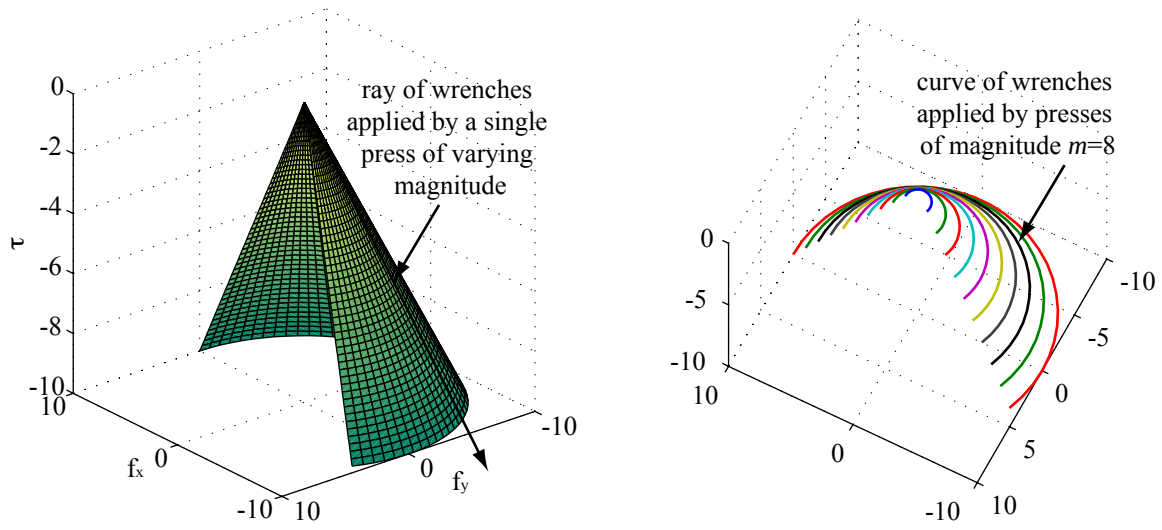


Figure 40: Assembly that can only be stably pressed in two places. The curved edge is a segment of a curve known as a circle involute, shown in more detail in figure 39. Any frictionless normal press of magnitude k along this edge applies the same torque.



(a) Surface in wrench space.

(b) Constant-magnitude curves of surface.

Figure 41: Surface of wrenches that can be applied along the curved edge of the assembly in figure 40. The surface is a circular cone, which is represented here both as a surface and as a collection of curves corresponding to presses of constant integer magnitude.

5.5.1 Gravity is known

As in section 5.3, we begin by considering the case that gravity is zero, and that we apply frictionless normal presses to the assembly surface. Given the latter assumption, we can only test points in wrench space that lie on the surface of applicable forces. Thus, the edges of the known stability cone must intersect points on this surface. Because of this, we can construct an assembly for which no press gives us stability information about any other presses we can apply.

Theorem 5 *There exist surfaces such that the set of stable pushes on the surface is discrete.*

Proof: Consider the frictionless assembly in figure 40, and let $\mathbf{G} = \mathbf{0}$. The parametric equation of the object's curved edge is

$$\mathbf{r}(s) = \begin{bmatrix} \sin(s) - s\cos(s) \\ \cos(s) + s\sin(s) \end{bmatrix}, 0 \leq s \leq \frac{9}{8}\pi, \tag{82}$$

and the transpose of the system Jacobian which relates presses to press wrenches is

$$\mathbf{J}^T = \begin{bmatrix} -1 & 0 \\ 0 & -1 \\ -1 & -1 \end{bmatrix}. \tag{83}$$

The surface of applicable wrenches is described by

$$\mathbf{w}(m, s) = m \begin{bmatrix} \cos(s) \\ -\sin(s) \\ -1 \end{bmatrix}, \tag{84}$$

where m is the magnitude of the press. The force-balance equations give us

$$\begin{bmatrix} -1 & 0 \\ 0 & -1 \\ -1 & -1 \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} = \begin{bmatrix} \cos(s) \\ -\sin(s) \\ -1 \end{bmatrix}, \tag{85}$$

Table 3: Stability of assemblies problem space.

Information	$\mathbf{G} = \mathbf{0}$		$\mathbf{G} \neq \mathbf{0}$		$\mathbf{G}$ unknown	
Press Type	Frictionless	Frictional	Frictionless	Frictional	Frictionless	Frictional
Edge Model						
Line	■	×	■	×	■	×
Curve	■	×	×	×	×	×
Plane	×	×	×	×	×	×
Gen. Surf.	×	×	×	×	×	×

■ - Result Given × - Open Problem

which only has a solution for values of s that satisfy

$$\sin(s) = 1 + \cos(s). \tag{86}$$

On the interval $s \in [0, \frac{9}{8}\pi]$, $\sin(s)$ and $1 + \cos(s)$ intersect only at $s = \frac{\pi}{2}, \pi$, thus only the two pushes

$$P_1 = \begin{bmatrix} 0 \\ -1 \end{bmatrix} \qquad \text{and} \qquad P_2 = \begin{bmatrix} -1 \\ 0 \end{bmatrix} \tag{87}$$

are stable. ■

This result has a geometric interpretation as well. Consider the surface of applied wrenches for this assembly, which is illustrated in figures 41(a) and 41(b). For constant m, $\mathbf{w}(m, s)$ is an arc of a circle in the $\tau = -m$ plane. It is simple to show that a polyhedral convex cone defined by any set of points on the curve does not contain any other points on the curve. Thus, no other presses on the object lie in the known stability cone, but any magnitude of a known stable press is stable.

5.6 Conclusion and open problems

In this section, we have discussed the assembly stability problem as a multibody stability problem, with the goal of finding stable regions along the surface of an assembly. And indeed, the results presented here are interesting when applied to this application, given the relatively small amount of related work in multibody grasping and stability problems. In addition, given the usual complexity of problems of this sort, our results are particularly noteworthy since they introduce a major simplification by ignoring geometry beyond the surface of the structure. There remain some limitations, however. We require that every object along the surface for which we wish to find a stability profile must be pressed separately. Also, finding parametric equations for real-world surfaces is nontrivial, although if we consider man-made assemblies, such as factory-built structures, or rubble from collapsed buildings, our results extended to polyhedral models will likely prove useful.

Another interesting application we might consider is a two-block assembly problem such as fixturing. For example, can we use pressing to determine whether an insertion task is complete, *i.e.* will this peg move further into this hole, or has it gone as far as it can go? For this problem, we seek to find remaining unstable pushes, and our results could be used to determine what region of applicable wrench space remains that has not yet been shown to be stable.

In the following sections, we discuss remaining open problems.

5.6.1 Classes of assembly

In section 5.2, we mapped out the problem space as shown in table 3. We have presented theoretical and experimental results for surfaces with polygonal edges, and theoretical results for edges that can be modeled as smooth parametric curves. In this section, we discuss areas where these results could be extended further.

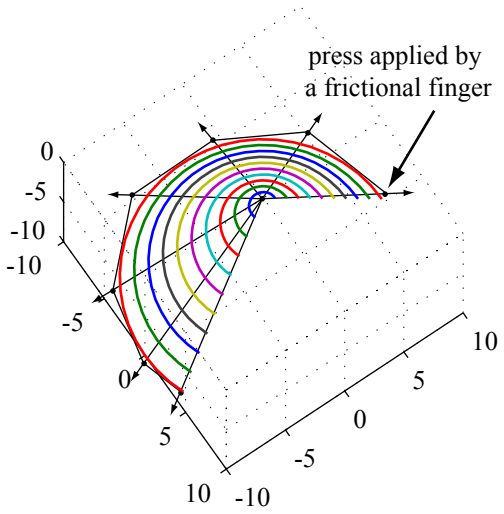


Figure 42: Surface of applicable frictionless wrenches contained in a known stability cone found by testing with frictional presses.

$\mathbf{G} = \mathbf{0}$, presses have friction. If, for the assembly given in figure 40 we can test wrenches that are not on the surface of applicable wrenches, we can define a cone in wrench space that contains the entire surface, as shown in figure 42. We can test these wrenches if we are allowed to press with a frictional finger. In this case, the set of applicable wrenches is a volume, not a surface, and depending on the coefficient of friction, we can at least contain the surface of wrenches applied normal to the surface. How we might select these points is left as an open question: possible methods might include picking points on the smallest regular polyhedral convex cone that contains the surface of frictionless presses, or finding local maxima in the curvature of the frictionless curve, and selecting wrenches where lines tangent to the curve at these points intersect.

$\mathbf{G} = \mathbf{0}$, presses have friction, surface of applied wrenches is not strictly convex. If the surface in wrench space is not strictly convex (or strictly concave), we can contain it in a known stability cone through some set of frictionless and frictional presses. Again, we leave the method for choosing the optimal points on and around the curve as an open problem. It is interesting to ask how being able to select points on the surface of normal wrenches might change the selection method.

Remaining cases. Additional open problems include how to select press points when the surface is translated by a known gravity wrench, how to select these points when gravity is unknown, and importantly, how to apply these principles to three-dimensional assemblies.

5.6.2 Selecting presses on an edge

Another open problem is to identify a method for determining the maximum area of an assembly's surface profile in the fewest possible presses. A related question is how to formalize the notion of information content in a press. In previous sections, we have presented results which might be used to devise an optimal strategy for selecting presses. In this section we briefly discuss some characteristics of such an algorithm for a planar assembly.

First, we must find, if they exist, a stable press and an unstable press on each side of it, in as few presses as possible. One approach is to distribute our samples evenly by testing the endpoints of the edge, and testing the midpoint of any region with unstable endpoints until a stable press is found. If we assume that every object in the assembly has a stable region that covers more than k% of its surface, this is a constant-time operation per surface object, so takes $O(n)$ time to test the assembly, where n is the number of objects at the surface of the assembly.

Once we have found a stable press on a given object, we must find the boundaries of the stable region. We may do so by employing a binary search on either side of the stable press, to find the endpoints of the stable region to the desired precision. This takes $O(L/p)$, where L is the length of the object, and p is the precision we seek.

What must be learned about the interior of the stable region depends strongly on the requirements of the specific application for which the stability profile is being determined. For some applications, we may be concerned with finding some interval on which we can apply a constant force. For others, we may seek the point at which we can apply the maximum force. Others, such as finding a force that can be applied to unjam a part during insertion, may seek only the unstable presses that remain on the surface of an object. All of these are left as open problems.

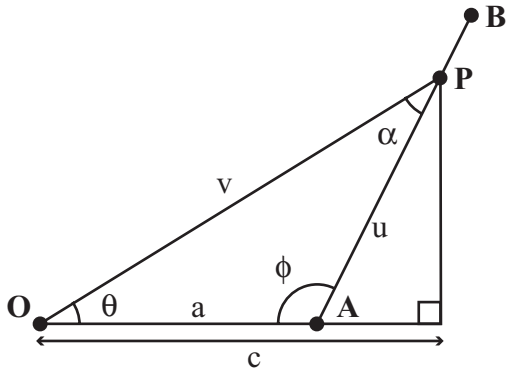


Figure 43: Simplified view of stable triangle in wrench space.

A Appendix: Law of sines proof of theorem 4

An alternative to solving for the point of intersection is to use trigonometric identities to solve for c directly. Consider $\triangle\mathbf{OAP}$ in figure 30. We know the length of side $\overline{\mathbf{OA}}$ and the slope of side $\overline{\mathbf{OP}}$. As previously discussed, c is the f_y-projection of the vector $\mathbf{P}$. A simplified view of the triangle from figure 30 is shown in figure 43. $\triangle\mathbf{OAP}$ has sides of length a, u, v, and opposing angles α, θ, ϕ. Given x_c as the slope of side v,

$$\sin\theta = \frac{x_c}{\sqrt{1+x_c^2}}, \qquad \cos\theta = \frac{1}{\sqrt{1+x_c^2}}. \tag{88}$$

Thus,

$$c = \frac{v}{\sqrt{1+x_c^2}}. \tag{89}$$

By the Law of Sines,

$$c = \frac{a\sin\phi}{\sin\alpha\sqrt{1+x_c^2}}. \tag{90}$$

We can find $\sin\phi$ and $\cos\phi$ by taking cross- and dot-products of the vectors $\mathbf{B}-\mathbf{A}$ and $\mathbf{O}-\mathbf{A}$. Let $D = ||\mathbf{B}-\mathbf{A}|| = \sqrt{(b-a)^2+(bx_b)^2}$.

$$(\mathbf{B}-\mathbf{A})\times(\mathbf{O}-\mathbf{A}) = ||\mathbf{B}-\mathbf{A}||\,||\mathbf{O}-\mathbf{A}||\sin\phi \tag{91}$$

$$\begin{bmatrix} b-a \\ bx_b \end{bmatrix} \times \begin{bmatrix} -a \\ 0 \end{bmatrix} = aD\sin\phi \tag{92}$$

$$\sin\phi = \frac{bx_b}{D} \tag{93}$$

$$(\mathbf{B}-\mathbf{A})\cdot(\mathbf{O}-\mathbf{A}) = ||\mathbf{B}-\mathbf{A}||\,||\mathbf{O}-\mathbf{A}||\cos\phi \tag{94}$$

$$(b-a)(-a) = aD\cos\phi \tag{95}$$

$$\cos\phi = \frac{a-b}{D} \tag{96}$$

Finally, since $\alpha = \pi - (\theta+\phi)$, $\sin\alpha = \sin(\theta+\phi)$. Let $E = \sqrt{1+x_c^2}$. Making the appropriate substitutions,

$$\sin\alpha = \sin\theta\cos\phi + \cos\theta\sin\phi \tag{97}$$

$$= \frac{x_c}{E}\frac{a-b}{D} + \frac{1}{E}\frac{bx_b}{D} \tag{98}$$

$$= \frac{x_c(a-b)+bx_b}{DE} \tag{99}$$

Substituting these values into equation 90, we find the same function for c as before,

$$c = \frac{abx_b}{bx_b + x_c(a-b)}. \tag{100}$$

B Appendix: Templates

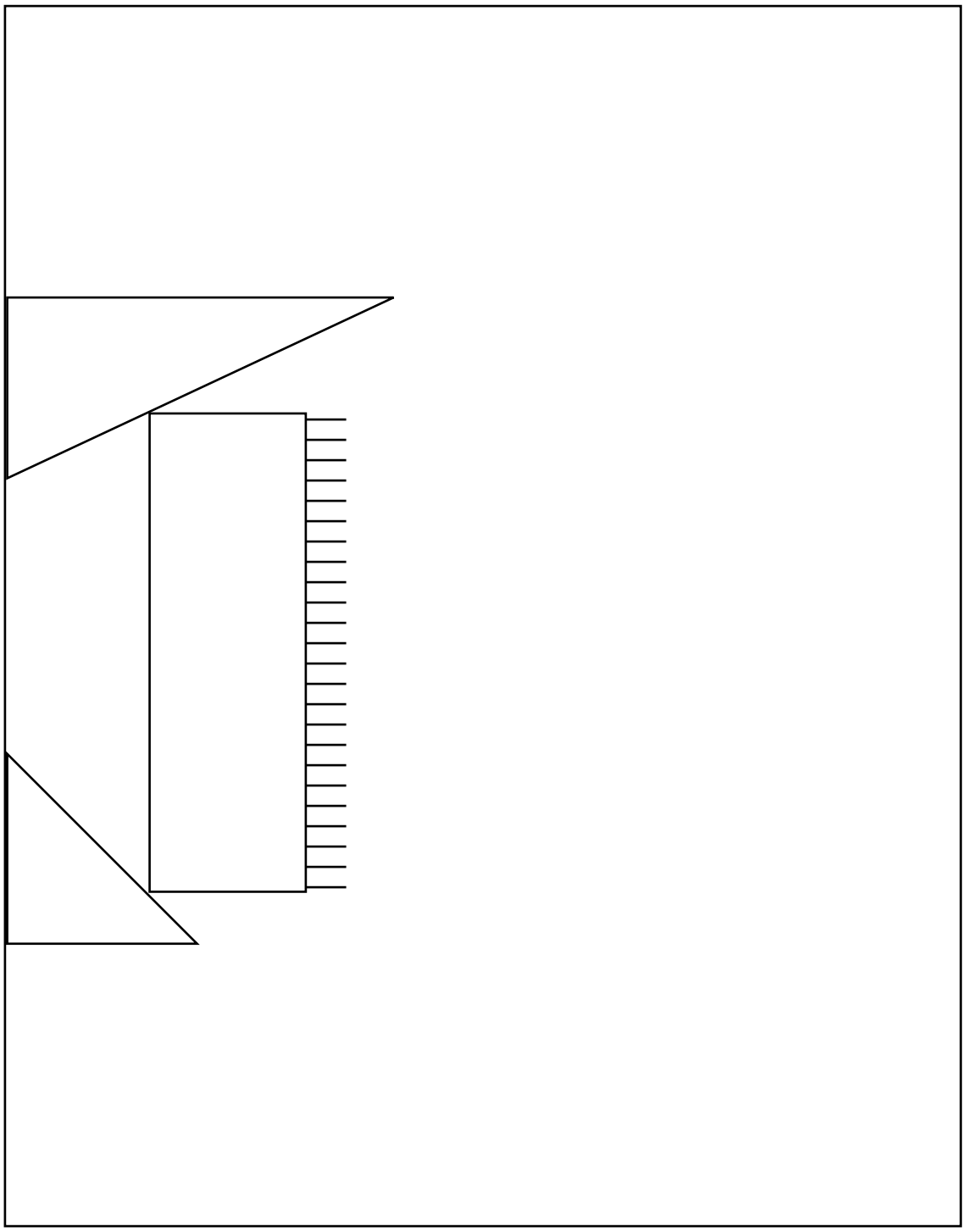


Figure 44: Template used to set up experiment 1.

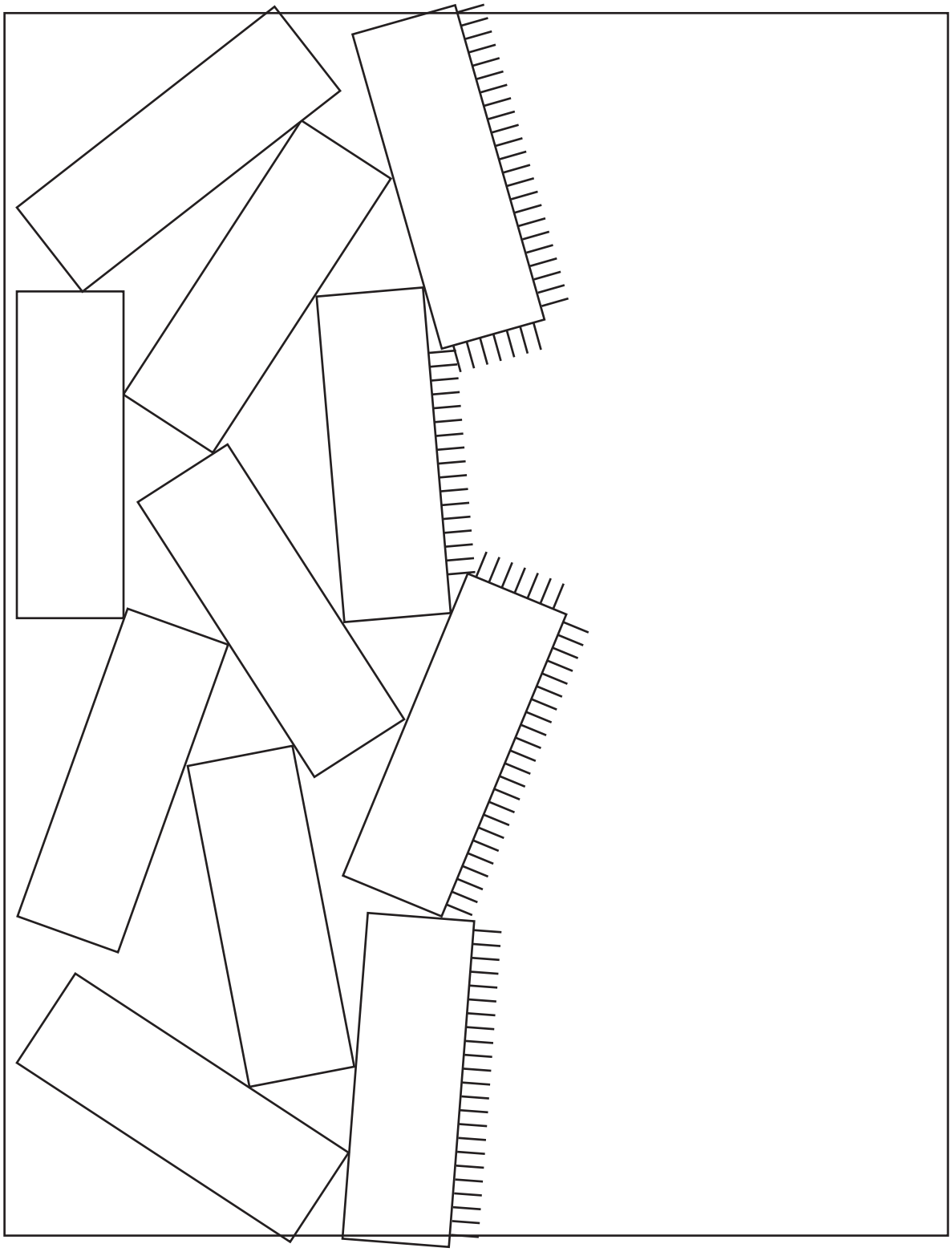


Figure 45: Template for structure shown figure 26. Template has been reduced in size to fit page margins.

References

[1] M. Anitescu and F. Potra. Formulating multi-rigid-body contact problems with friction as solvable linear complementarity problems. *ASME Journal of Nonlinear Dynamics*, 14:231–247, 1997.

[2] M. Anitescu, F. A. Potra, and D. E. Stewart. Time-stepping for three-dimensional rigid body dynamics. *Computer Methods in Applied Mechanics and Engineering*, 177:183–197, 1999.

[3] V. Ayyadevara, D. Bourne, K. Shimada, and R. Sturges. Interference-free polyhedral configurations for stacking. *IEEE Transactions on Robotics and Automation*, 18(2):147 – 165, April 2002.

[4] D. J. Balkcom and J. C. Trinkle. Computing wrench cones for planar contact tasks. *International Journal of Robotics Research*, pages 1053–1066, 2002.

[5] D. Baraff. Issues in computing contact forces for non-penetrating rigid bodies. *Algorithmica*, 10:292–352, 1993.

[6] D. Baraff. Fast contact force computation for nonpenetrating rigid bodies. In *SIGGRAPH*, pages 23–34, 1994.

[7] D. Baraff. Linear-time dynamics using Lagrange multipliers. In *SIGGRAPH*, pages 137–146, Aug. 1996.

[8] D. Baraff and A. Witkin. Physically based modeling, 2001. Online SIGGRAPH 2001 Course Notes.

[9] M. Berkelaar, K. Eikland, and P. Notebaert. lp_solve, open source (mixed-integer) linear programming system, May 2004. Version 5.1.0.0.

[10] J. D. Bernheisel and K. M. Lynch. Stable transport of assemblies: Pushing stacked parts. *IEEE Transactions on Automation Science and Engineering*, 1(2):163–168, October 2004.

[11] J. D. Bernheisel and K. M. Lynch. Stable pushing of assemblies. In *IEEE International Conference on Robotics and Automation*, Barcelona, Spain, April 2005.

[12] M. Blum, A. Griffith, and B. Neumann. A stability test for configurations of blocks. Technical Report AIM-188, Massachusetts Institute of Technology, February 1970.

[13] N. Bonschanscher, H. van der Drift, S. J. Buckley, and R. H. Taylor. Subassembly stability. In *AAAI 88: The seventh national conference on Artificial Intelligence*, volume 2, pages 780–785, Aug. 1988.

[14] J.-S. Cheong, K. Goldberg, M. H. Overmars, and A. F. van der Stappen. Immobilizing hinged polygons. In *icra*, pages 876–881, 2002.

[15] R. W. Cottle and G. B. Dantzig. Complementary pivot theory of mathematical programming. *Linear algebra and its applications*, 1:103–125, 1968.

[16] R. W. Cottle, J.-S. Pang, and R. E. Stone. *The Linear Complementarity Problem*. Academic Press, 1992.

[17] G. B. Dantzig. *Linear Programming and Extensions*. Princeton University Press, 1963.

[18] G. B. Dantzig and W. Orchard-Hayes. The product form for the inverse in the simplex method. *Mathematical Tables and Other Aids to Computation*, 8:183–195, 1954.

[19] S. Dirkse and M. Ferris. The PATH solver: a non-monotone stabilization scheme for mixed complementarity problems. *Optimization Methods and Software*, 5:123–156, 1995.

[20] T.-H. Eng, Z.-K. Ling, W. Olson, and C. McLean. Feature-based assembly modeling and sequence generation. *Computers & industrial engineering*, 36(1):17–33, Jan. 1999.

[21] M. A. Erdmann. On motion planning with uncertainty. Master's thesis, Massachusetts Institute of Technology, Aug. 1984.

[22] R. Featherstone. *Robot Dynamics Algorithms*. Kluwer Academic Publishers, 1987.

[23] J. Forrest and J. Tomlin. Updating triangular factors of the basis to maintain sparsity in the product form simplex method. *Mathematical Programming*, 2:263–278, 1972.

[24] R. Fourer. Linear programming frequently asked questions, 2000. http://www-unix.mcs.anl.gov/otc/Guide/faq/linear-programming-faq.html.

[25] A. Greenfield, U. Saranli, and A. A. Rizzi. Solving models of controlled dynamic planar rigid-body systems with frictional contact. *International Journal of Robotics Research*, 24(11):911–931, Nov. 2005.

[26] E. Guendelman, R. Bridson, and R. Fedkiw. Nonconvex rigid bodies with stacking. *SIGGRAPH*, pages 871–878, 2003.

[27] S. K. Hargrove and A. Kusiak. Computer-aided fixture design: a review. *International Journal of Production Research*, 32(4):733–753, Apr. 1994.

[28] C. J. M. Heemskerk. The use of heuristics in assembly sequence planning. *Annals of the CIRP*, 38(1):37–40, 1989.

[29] R. Hoffman. Automated assembly planning for B-rep products. In *Systems Engineering, 1990., IEEE International Conference on*, pages 391–394, Pittburgh, PA, USA, Aug. 1990.

[30] N. Karmarkar. A new polynomial-time algorithm for linear programming. In *STOC '84: Proceedings of the sixteenth annual ACM symposium on Theory of computing*, pages 302–311, New York, NY, USA, 1984. ACM Press.

[31] K. Kotay, D. Rus, and M. Vona. Using modular self-reconfiguring robots for locomotion. In *Proceedings of Experimental Robotics IV*, 2000.

[32] C. Lemke. Bimatrix equilibrium points and mathematical programming. *Management Science*, 11(7), May 1965.

[33] A. Loomis. Computation reuse in stacking and unstacking. Technical Report TR2005-563, Dartmouth College, Computer Science, Hanover, NH, November 2005.

[34] P. Lötstedt. Coulomb friction in two-dimensional rigid-body systems. *Zeitschrift für Angewandte Mathematik und Mechanik*, 61:605–615, 1981.

[35] P. Lötstedt. Numerical simulation of time-dependent contact friction problems in rigid body mechanics. *SIAM Journal of Scientific Statistical Computing*, 5(2):370–393, 1984.

[36] M. T. Mason. *Mechanics of robotic manipulation*. MIT Press, 2001.

[37] M. T. Mason and Y. Wang. On the inconsistency of rigid-body frictional planar mechanics. In *IEEE International Conference on Robotics and Automation*, pages 524–528, 1988.

[38] R. Mattikalli, D. Baraff, P. Khosla, , and B. Repetto. Gravitational stability of frictionless assemblies. *IEEE Transactions on Robotics and Automation*, 11(3):374–388, June 1995.

[39] R. Mattikalli, D. Baraff, and P. Khosla. Finding all stable orientations of assemblies with friction. *IEEE Transactions on Robotics and Automation*, 12(2):290–301, April 1996.

[40] L. S. H. D. Mello and A. C. Sanderson. And/or graph represenation of assembly plans. *IEEE Transactions on Robotics and Automation*, 6(2):188–199, Apr. 1990.

[41] L. S. H. D. Mello and A. C. Sanderson. A correct and complete algorithms for the generation of mechanical assembly sequences. *IEEE Transactions on Robotics and Automation*, 7(2):228–240, Apr. 1991.

[42] V. J. Milenkovicy and H. Schmidlz. Optimization-based animation. In *SIGGRAPH*, pages 37–46, 2001.

[43] B. Mishra, J. T. Schwartz, and M. Sharir. On the existence and synthesis of multifinger positive grips. *Algorithmica*, 2(4):541–558, 1987.

[44] B. Mishra and M. Teichmann. On immobility. In *Special Issue: Robot Kinematics, Laboratory Robotics and Automation: International Journal*, volume 4, pages 145–153. VCH Publisher, 1992.

[45] M. D. P. Monteiro Marques. *Differential Inclusions in Nonsmooth Mechanical Problems: Shocks and Dry Friction*, volume 9 of *Progress in Nonlinear Differential Equations and Their Applications*. Birkhauser Verlag, Basel, Boston, Berlin, 1993.

[46] J. Moreau. Frottement, adhésion, lubrification. *Comptes Rendus, Série II*, 302:799–801, 1986.

[47] J. Moreau. Unilateral contact and dry friction in finite freedom dynamics. In J. Moreau and P. Panagiotopoulos, editors, *Nonsmooth Mechanics and Applications*, CISM Courses and Lectures #302, pages 1–82, Vienna, New York, 1988. Springer-Verlag.

[48] H. Mosemann, F. Röhrdanz, and F. M. Wahl. Stability analysis of assemblies considering friction. *IEEE Transactions on Robotics and Automation*, 13(6):805–813, December 1997.

[49] K. G. Murty. *Linear complementarity, linear and nonlinear programming*. Heldermann, Berlin, 1988.

[50] V.-D. Nguyen. Constructing force-closure grasps. *International Journal of Robotics Research*, 7(3), 1988.

[51] P. Painlevé. Sur les lois du frottemont de glissement. *C. R. Acad. Sci.*, 121:112–115, 1895.

[52] R. S. Palmer. *Computational complexity of motion and stability of polygons*. PhD thesis, Cornell university, Ithaca, NY, 1987.

[53] J.-S. Pang and D. E. Stewart. A unified approach to discrete frictional contact problems. *International Journal of Engineering Science*, 37(13):1747–1768, Oct. 1999.

[54] J.-S. Pang and J. Trinkle. Stability characterizations of rigid body contact problems with Coulomb friction. *Zeitschrift für Angewandte Mathematik und Mechanik*, 80(10):643–663, 2000.

[55] M. Paterson and U. Zwick. Overhang. In *SODA '06: Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 231–240, New York, NY, USA, 2006. ACM Press.

[56] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cabridge University Press, second edition, 1992.

[57] E. Rimon and J. Burdick. Mobility of bodies in contact, i: A new 2^{nd} order mobility index for multiple-finger grasps. In *IEEE International Conference on Robotics and Automation*, volume 3, pages 2329–2335, 1994.

[58] E. Rimon and J. Burdick. On force and form closure for multiple finger grasps. In *IEEE International Conference on Robotics and Automation*, volume 2, pages 1795–1800, Apr. 1996.

[59] B. Romney. Atlas: an automatic assembly sequencing and fixturing system. In *Theory and Practice of Geometric Modeling*, pages 397–415, 1997.

[60] B. Romney. *On the concurrent design of assembly sequences and fixtures*. PhD thesis, Stanford University, March 1997.

[61] S. Singh and S. Simmons. Task planning for robotic excavation. In *Proc. IEEE/RSJ International Conference on Intelligent Robot Systems*, July 1992.

[62] R. Smith. Open dynamics engine: v0.5 user guide, May 2004.

[63] J. Snoeyink and J. Stolfi. Objects that cannot be taken apart with two hands. In *SCG '93: Proceedings of the ninth annual symposium on Computational geometry*, pages 247–256, New York, NY, USA, 1993. ACM Press.

[64] D. Stewart and J. Trinkle. An implicit time-stepping scheme for rigid body dynamics with inelastic collisions and Coulomb friction. *International Journal of Numerical Methods in Engineering*, 39:2673–2691, 1996.

[65] D. E. Stewart. Rigid-body dynamics with friction and impact. *SIAM Review*, 42(1):3–39, 2000.

[66] J. Trinkle. private communication, 2006. Rensselaer Polytechnic Institute, NY.

[67] J. Trinkle, J.-S. Pang, S. Sudarsky, and G. Lo. On dynamic multi-rigid-body contact problems with Coulomb friction. *Zeitschrift für Angewandte Mathematik und Mechanik*, 77(4):267–279, 1997.

[68] J. Trinkle, J. Tzitzouris, and J. Pang. Dynamic multi-rigid-body systems with concurrent distributed contacts: theory and examples. *Philosophical transactions on mathematical, physical, and engineering sciences*, 359(1789):2575–2593, 2001.

[69] C. Unsal and P. K. Khosla. Solutions for 3-d self-reconfiguration in a modular robotic system: Implementation and motion planning. In *Proceedings of SPIE, Sensor Fusion and Decentralized Control in Robotic Systems III*, November 2000.

[70] R. J. Vanderbei. *Linear Programming: Foundations and Extensions*. Kluwer Academic Publishers, 1996.

[71] R. H. Wilson, L. Kavraki, J.-C. Latombe, and T. Lozano-Pérez. Two-handed assembly sequencing. *International Journal of Robotics Research*, 14(4):335–350, 1995.

[72] R. H. Wilson and J.-C. Latombe. Geometric reasoning about mechanical assembly. *Artifical Intelligence*, 71:371–396, 1994.

[73] R. H. Wilson and J.-F. Rit. Maintaining geometric dependencies in an assembly planner. In *IEEE International Conference on Robotics and Automation*, May 1990.

[74] A. Witkin and W. Welch. Fast animation and control of nonrigid structures. In *Computer Graphics (Proc. SIGGRAPH)*, volume 24, pages 243–252, New York, August 1990.

[75] U. Zwick. Jenga. In *SODA*, pages 243–246, 2002.