The Grid Bracing Problem and a Generalization

by

Scott T. Laine

A Thesis

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Master of Science

in

Applied Mathematics

by

May 2006

APPROVED:

Professor Dr. Brigitte Servatius, Major Thesis Advisor

Professor Dr. Bogdan Vernescu, Head of Department

Abstract

The standard grid bracing problem has a nice solution via the brace graph. If we introduce a window by removing an interior vertex of the grid, this solution comletely breaks down. We examine a 6 x 10 unit grid with a 2 x 2 window and provide an optimal solution via the Rigidity Matrix.

Acknowledgements

I really could not have done this without the support of most importantly my family: Thom, Lynda, and Krysten, and my friends. The significant help of my advisor, Dr. Brigitte Servatius, who was always more than giving of her time and efforts. (I appreciate you letting me harass you at your house.) My girlfriend who put up with my constant stress, thanks Taryn. As well as my classmates: Joseph Simonis, Elizabeth Teixeira, Joseph Spino, Rebecca Wasyk, Ashley Moraski, Criselda Toto. Each one helped academically as well as psychologically to help me get through it all. Colleen Lewis and Severino V. Gervacio for there extensive knowledge and assistance with LaTex. There are several others I have failed to mention since this is a finite list but I am truly thankful to have these people around me who were so gracious with there time and assistance. Thank You.

Contents

List of Figures

Chapter 1

Introduction

1.1 Graphs

The basic concepts from graph theory are defined in standard texts, e.g. Graver [2], Lovasz [3], Graver and Servatius [1], and Rosen [4]. An *edge* is a line segment joining two vertices,denoted by its endpoints. A *graph*, G(V,E), consists of a finite set of *vertices* denoted V(G), a finite set of *edges* denoted E(G), and an assignment of an unordered pair of elements of V(G) to each edge e $\in$ E(G) called the *endpoints*, of e. A graph G, (V,E), is said to be *bipartite* if it has a partition of its vertex set into two sets, A and B (V = A $\cup$ B, A $\cap$ B = $\emptyset$), so that every edge in E has one endpoint in each set.

The power and application of graphs is in the way they show relationships, as suggested by the example given by Graver [2]. Suppose we let W denote the five workers in a small carpentry shop that makes table legs, $W = \{a,b,c,d,e\}$ for Al, Betty, Cliff, Dan, and Ethel; and suppose we let S denote the four skills needed in the process of making table legs, $S = \{f,s_1,s_2,t\}$ for finishing, sawing, sanding, and turning (on a lathe). We can model the relationship between workers and tasks by

making a list of pairs consisting of a worker and a task he or she can perform. For instance, assume that

$$E=\{a,f\},\{a,s_1\},\{b,s_1\},\{c,s_1\},\{c,s_1\},\{c,s_2\},\{c,t\},\{d,s_2\},\{e,s_1\},\{e,t\}$$

is a complete listing of such pairs. The two sets $V = W \cup S$ and E encode all of the information about 'who can do what' in our little carpentry shop. To get a 'picture' of this information we select nine points in the plane, label them by the elements of V and draw in a line segment for each pair in E. This yields Figure 1.1:

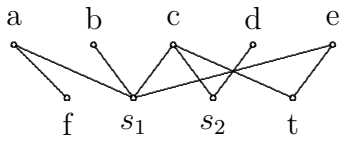


Figure 1.1: Carpentry Shop

An *embedding* of a graph $\{V, E\}$ is a function $\mathbf{p}$ from a vertex set into m-space, $\mathbf{p}$:V $\rightarrow R^m$. A graph, $\{V, E\}$, combined with an embedding is called a *framework* ,$\{V, E, \mathbf{p}\}$. A framework embedded in the plane has for each vertex an assignment of values (x,y) to indicate location within the plane. For the purposes of this paper, frameworks will be restricted to 2-space, the Euclidean plane.

1.2 Connectivity

Connectivity will be developed by first considering a *disconnected* graph. A graph (V,E) is said to be *disconnected* if the vertex set can be partitioned into two nonempty sets A and B (V = A $\cup$ B, A $\cap$ B = $\emptyset$, A $\neq \emptyset$, B $\neq \emptyset$) so that no edge has one endpoint in A and the other endpoint in B. We say that a graph (V,E) is *connected* if no such partition exists. A *path* between vertices a and b is a sequence of vertices $a_0, a_1, ..., a_k$ where,

1. $a = a_0$ and $a_k = b$;

2. $a_0, a_1, ..., a_k$ are distinct;

3. a_{i-1} and a_i are adjacent, for i = 1,...,k.

Two vertices are *adjacent*, if the vertices share an edge. The edges joining successive vertices in the sequence are called the *edges of the path*, and the number of these edges is called the *length* of the path.

Theorem 1 *A graph (V,E) is connected if and only if every pair of its vertices is joined by a path.*

Proof. Assume that there are two vertices a and b in V that are not joined by a path. Let A be the set of all vertices c so that there is a path from a to c and let B = V - A. Since $a \in$ A and $b \in$ B, A and B partition V into two nonempty sets. Suppose that there is an edge $e = \{a', b'\}$ with $a' \in$ A and $b' \in$ B. Then there must exist a path from a to a' and the path (of length 1) from a' to b' yield a path from a to b'. But, this contradicts the fact that b' is not in A. We conclude that there is no edge with one endpoint in A and the other in B; hence (V,E) is not connected.

Now assume that every two vertices of (V,E) are joined by a path and let A and B be any partition of V into two nonempty sets. Choose $a \in$ A and $b \in$ B and let $a = a_0, a_1, ..., a_k = b$ be a path joining a and b. Let a_i be the last vertex along the path in A; since $b \in$ B, $i < k$. Then the next vertex a_{i+1} is in B and the edge $\{a_i, a_{i+1}\}$ has one endpoint in A and the other in B. We conclude that (V,E) is connected. Q.E.D.

Chapter 2

Rigidity Theory

2.1 Motions

Let (V, E, $\mathbf{p}$) be a framework with indexed vertex set $\mathrm{V} = \{a_1, ..., a_n\}$. Then:

- A *motion* of framework F comprises an indexed family of functions $\mathbf{P}_i$: $[0,1] \to \mathbb{R}^m$, i = 1,...,n, so that:

  1. $\mathbf{P}_i(0) = \mathbf{p}_i$, for all i;
  2. $\mathbf{P}_i(t)$ is differentiable on the interval $[0,1]$, for all i;
  3. $|\mathbf{P}_i(t), \mathbf{P}_j(t)| = \left|\mathbf{p}_i, \mathbf{p}_j\right|$, for all $t \in [0,1]$ and $\{a_i, a_j\} \in$ E.

- The function $\mathbf{P}_i(t)$ is called the *trajectory* of the point $\mathbf{p}_i$ under the motion.

- The notation $\left|\mathbf{p}_i, \mathbf{p}_j\right|$ denotes the distance between the points $\mathbf{p}_i$ and $\mathbf{p}_j$ in $\mathbb{R}^m$. We have

  - $\left|\mathbf{p}_i, \mathbf{p}_j\right| = |x_i - x_j| = \sqrt{(x_i - x_j)^2}$, if m = 1;
  - $\left|\mathbf{p}_i, \mathbf{p}_j\right| = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$, if m = 2;

- For a fixed time t, the framework (V,E,$\mathbf{q}$), where $\mathbf{q}_i = \mathbf{P}_i(t)$, is the position to which the initial framework has moved at time t.

- A motion $\{\mathbf{P}_i\}$ of the framework (V,E,$\mathbf{p}$) is a *rigid motion* if all the distances between vertices are preserved by the motion:

$$|\mathbf{P}_i(t), \mathbf{P}_j(t)| = \left|\mathbf{p}_i, \mathbf{p}_j\right|,$$

  for all $t \in [0, 1]$ and all $1 \leq i < j \leq n$.

- A motion $\mathbf{P}$ of the framework (V,E,$\mathbf{p}$) is a *deformation* if the distance between at least one pair of vertices is changed by the motion:

$$|\mathbf{P}_i(t), \mathbf{P}_j(t)| \neq \left|\mathbf{p}_i, \mathbf{p}_j\right|,$$

  for some $t \in [0, 1]$ and some $\{a_i, a_j\} \neq E$

- A framework (V,E,$\mathbf{p}$) is said to be *rigid* if all of its motions are rigid motions, that is, if it admits no deformations.

2.2 Velocity Vector

The differentiability of a motion $\mathbf{P}_i$ implies that, at each position along the trajectory of a point, the velocity vector is well defined; For a vertex $a_i \in V$, the trajectory of the corresponding point of the framework is given by $\mathbf{P}_i(t)$. If we let $x_i(t)$ and $y_i(t)$ denote the coordinates of this given point at time k, the *velocity vector* which we denote by $\mathbf{P}_i^{'}(t)$, is given by

$$(x^{'}(t), y^{'}(t)) = (\tfrac{d}{dt}x(t), \tfrac{d}{dt}y(t)).$$

2.3 Infinitesimal Motion

- An *infinitesimal motion* of the m-dimensional framework (V,E,$\mathbf{p}$) is a function $\mathbf{q} : V \to \mathbb{R}^m$ so that $(\mathbf{p}_i - \mathbf{p}_j) \cdot (\mathbf{q}_i - \mathbf{q}_j) = 0$, for all $\{a_i, a_j\} \in E$.

- An infinitesimal motion $\mathbf{q}$ of the framework (V,E,$\mathbf{p}$) is called an *infinitesimal rigid motion* if $(\mathbf{p}_i - \mathbf{p}_j) \cdot (\mathbf{q}_i - \mathbf{q}_j) = 0$, for all $a_i, a_j \in V$.

- An infinitesimal motion $\mathbf{q}$ of the framework (V,E,$\mathbf{p}$) is an *infinitesimal deformation* if $(\mathbf{p}_i - \mathbf{p}_j) \cdot (\mathbf{q}_i - \mathbf{q}_j) \neq 0$, for some $a_i, a_j \in V$.

- A framework (V,E,$\mathbf{p}$) is said to be *infinitesimally rigid* if all of its infinitesimal motions are infinitesimal rigid motions.

- If a framework (V,E,$\mathbf{p}$) is infinitesimally rigid then it is rigid.

2.4 Mathematical Framework

Let (V,E,$\mathbf{p}$) denote an arbitrary but fixed framework in 2-dimensional space, let V $= \{a_1, ..., a_n\}$, and (x_i, y_i) denote the embedding of $\mathbf{p}_i$, for i = 1,...,n. Let $\mathbf{q}$: V $\to \mathbb{R}^2$ be an arbitrary function from the vertex set into the vector space $\mathbb{R}^2$ and let (u_i, v_i) denote the coordinates of $\mathbf{q}_i$, for i = 1,...,n. The vector $\mathbf{q}_i$ is assigned to the point $\mathbf{p}_i$.

The set of vectors in $\mathbb{R}^{2n}$ that corresponds to the infinitesimal rigid motions of a framework is the solution set to the collection of linear equations corresponding to all pairs of distinct vertices in V. Also, the set of vectors in $\mathbb{R}^{2n}$ that corresponds to the infinitesimal rigid motions of a framework is the solution set to the collection of linear equations corresponding to the edges in E.

Let $\mathcal{M}(\mathcal{F})$ denote the *subspace of infinitesimal motions of the framework* $\mathcal{F}$ =

(V,E,**p**) and let $\mathcal{R}(\mathcal{F})$ denote *the subspace of infinitesimal rigid motions.* Since every infinitesimal rigid motion of $\mathcal{F}$ is an infinitesimal motion of $\mathcal{F}$,

$$\mathcal{R}(\mathcal{F}) \subseteq \mathcal{M}(\mathcal{F}).$$

So $\mathcal{F}$ is infinitesimally rigid if and only if

$$\mathcal{R}(\mathcal{F}) = \mathcal{M}(\mathcal{F}),$$

since $\mathcal{R}(\mathcal{F}) \subseteq \mathcal{M}(\mathcal{F})$, $\mathcal{F}$ is infinitesimally rigid if and only if

$$dim\left[\mathcal{R}(\mathcal{F})\right] = dim\left[\mathcal{M}(\mathcal{F})\right].$$

2.5 Infinitesimal Translation

Let $\tau_{(u,v)}$:$\mathbb{R}^2 \to \mathbb{R}^2$ be the infinitesimal translation that assigns the vector (u, v) to each point in the plane, that is, the constant function $\tau_{(u,v)}(x, y) = (u, v)$. If (x, y) and $(\hat{x}, \hat{y})$ are any two points in the plane,

$$(\hat{x} - x, \hat{y} - y) \cdot (\tau_{(u,v)}(\hat{x}, \hat{y}) - \tau_{(u,v)}(x, y)) = (\hat{x} - x, \hat{y} - y) \cdot (0, 0) = 0,$$

confirming that the infinitesimal translations neither shrink nor stretch the distance between any pair of points in the plane, therefore it is an infinitesimal rigid motion of the plane. The collection of all infinitesimal translations forms a 2-dimensional vector space spanned by $\tau_{(1,0)}$ and $\tau_{(0,1)}$. So for any infinitesimal translation $\tau_{(u,v)}$,

$$\tau_{(u,v)} = u\tau_{(1,0)} + v\tau_{(0,1)}.$$

2.6 Infinitesimal Rotations

Let (x, y) be any point in the plane, The trajectory of (x, y) under a rotation about the origin is given by

$$(x(t), y(t)) = (rcos(\theta + at), rsin(\theta + at)), \text{ where } x = x(0) = rcos(\theta)$$
$$y = y(0) = sin(\theta)$$

for time $t \in [0, 1]$. The rotation is counterclockwise when a is positive, and clockwise when a is negative. The larger the absolute value of a, the faster the rotation. Differentiating with respect to time and evaluating the derivative at $t = 0$,

$$a(rcos(\theta), -rsin(\theta)) = a(y, -x).$$

The initial velocity of a rotation at a point (x, y) is perpendicular to the segment from that point to the center of rotation, and its length is proportional to the distance from that point to that center.

A *unit infinitesimal rotation* with center (x_0, y_0),

$$\rho_{(x_0,y_0)}(x, y) = (y - y_0, x_0 - x).$$

The remaining infinitesimal rotations with center (x_0, y_0) are the scalar multiples of $\rho_{(x_0,y_0)}$. This is an infinitesimal rigid motion of the plane.

Let (x, y) and $(\hat{x}, \hat{y})$ be any two point in the plane,

$$\begin{aligned}
(\hat{x} - x, \hat{y} - y) \quad &\cdot \quad (\rho_{(x_0,y_0)}(\hat{x}, \hat{y}) - \rho_{(x_0,y_0)}(x, y)) \\
&= (\hat{x} - x, \hat{y} - y) \cdot ((\hat{y} - y_0, x_0 - \hat{x}) - (y - y_0, x_0 - x)) \\
&= (\hat{x} - x, \hat{y} - y) \cdot (\hat{y} - y, x - \hat{x}) \\
&= 0.
\end{aligned}$$

2.7 Infinitesimal Translations and Rotations

Theorem 2 *The set of all infinitesimal translations and rotations of the plane forms a 3-dimensional vector space.*

Proof $\tau_{(1,0)}, \tau_{(0,1)}$ and $\rho_{(0,0)}$ are independent infinitesimal motions of the plane. Any infinitesimal translation of the plane is a linear combination of $\tau_{(1,0)}$ and $\tau_{(0,1)}$. The next equation shows that any infinitesimal rotation of the plane is a linear combination of these three vectors:

$$a\rho_{(x_0,y_0)} = a\rho_{(0,0)} + (-ay_0)\tau_{(1,0)} + (ax_0)\tau_{(0,1)}.$$

Any linear combination of these three vectors is either an infinitesimal rotation or an infinitesimal translation. Moreover,

$$a\rho_{(0,0)} + b\tau_{(1,0)} + c\tau(0,1) = \tau_{(b,c)}, if a = 0;$$

$$a\rho_{(0,0)} + b\tau_{(1,0)} + c\tau(0,1) = a\rho_{(c/a,-b/a)}, if a \neq 0.$$

So $\rho_{(0,0)}$, $\tau_{(0,1)}$, and $\tau_{(1,0)}$ span the space of infinitesimal motions in the plane. To see that $\rho_{(0,0)}$, $\tau_{(0,1)}$, and $\tau_{(1,0)}$ form a basis, we observe that if,

$$(a\rho_{(0,0)} + b\tau_{(0,1)} + c\tau_{(1,0)})(x,y) = (x,y) \text{ for all } (x,y) \in \mathbb{R}^2 \text{ then } a = b = c = 0$$

2.8 General Position

A set P = $\{\mathbf{p}_1, ..., \mathbf{p}_n\}$ in the plane is in *general position* if no two points are equal and no three lie on a line. A framework (V,E,$\mathbf{p}$) is in *general position* whenever the set of points P = $\left\{\mathbf{p}_1, ..., \mathbf{p}_{|V|}\right\}$ is in general position.

Theorem 3 *Let $\mathcal{F}$ = (V,E,$\boldsymbol{p}$) be any framework embedded in the plane in general position with $|V| \geq 2$. Then $\mathcal{R}(\mathcal{F})$ is the 3-dimensional space of infinitesimal translations and rotations restricted to the points in $P = \{\boldsymbol{p}_1, ..., \boldsymbol{P}_n\}$.*

Proof.

Let $\mathbf{q} \in \mathcal{R}(\mathcal{F})$. Let V $= \{a_1, ..., a_n\}$ and denote the coordinates of $\mathbf{p}_i$ by (x_i, y_i) and the coordinates of $\mathbf{q}_i$ by (u_i, v_i), for all i. Clearly, the restriction of the translation $\tau_{(u_1,v_1)}$ to P is a rigid motion of $\mathcal{F}$. Hence $\hat{\mathbf{q}} = \mathbf{q} - \tau_{(u_1,v_1)} \in \mathcal{R}(\mathcal{F})$. Denote the coordinates of $\hat{\mathbf{q}}_i$ by $(\hat{u}_i, \hat{v}_i)$, for all i, and note that $\hat{\mathbf{q}}_1 = (0, 0)$. The condition

$$(x_2 - x_1, y_2 - y_1) \cdot (\hat{u}_2 - 0, \hat{v}_2 - 0) = 0$$

gives us that $\hat{\mathbf{q}}_2 = a(y_2 - y_1, x_1 - x_2)$, where $a = \hat{u}_2/(y_2 - y_1)$ when $y_1 \neq y_2$ and $a = \hat{v}_2/(x_1 - x_2)$ otherwise. (Since the points are in general position, either $y_1 \neq y_2$ or $x_1 \neq x_2$. This also is true for unit grids to be defined later.)

Now define

$$\check{\mathbf{q}} = \hat{\mathbf{q}} - a\rho_{(x_1,y_1)} = \mathbf{q} - \tau_{(u_1,v_1)} - a\rho_{(x_1,y_1)};$$

as above, $\check{\mathbf{q}} \in \mathcal{R}(\mathcal{F})$. Furthermore, $\check{\mathbf{q}}_1 = (0, 0)$ and $\check{\mathbf{q}}_2 = (0, 0)$. Finally, for $i > 2$, we must have

$$(x_i - x_1, y_i - y_1) \cdot (\check{u}_i - 0, \check{v}_i - 0) = 0$$

and

$$(x_i - x_2, y_i - y_2) \cdot (\check{u}_i - 0, \check{v}_i - 0) = 0.$$

Since (x_i, y_i) is not on the line through (x_1, y_2) and (x_2, y_2), the vectors $(x_i - x_1, y_i - y_1)$ and $(x_i - x_2, y_i - y_2)$ are not parallel, whereas $(\check{u}_i, \check{v}_i)$ is perpendicular to both of them. Hence the only solution to the above equations is $(\check{u}_i, \check{v}_i) = (0, 0)$. So $\check{\mathbf{q}}$ assigns the zero vector to each point of $\mathcal{F}$ and, therefore, $\mathbf{q}$ is the restriction of $\tau_{(u_1,v_1)} + a\rho_{(x_1,y_1)}$ to P.

Infinitesimal rigid motions must be a combination of an infinitesimal rigid motion since all motions of the plane are a linear combination of infinitesimal translations or rotations.

Since $\mathcal{R}(\mathcal{F})$ is a 3-dimensional subset of $\mathcal{M}(\mathcal{F})$, $\mathcal{M}(\mathcal{F}) = \mathcal{R}(\mathcal{F})$ if and only if $\mathcal{M}(\mathcal{F})$ is also 3-dimensional.

Theorem 4 *Let $\mathcal{F}$ = (V,E,**p**) be a framework embedded in the plane in general position with $|V| \geq 2$. Then $dim\,[\mathcal{M}(\mathcal{F})] \geq 3$, and $\mathcal{F}$ is infinitesimally rigid if and only if $dim\,[\mathcal{M}(\mathcal{F})] = 3$.*

2.9 Degrees of Freedom

Let $\mathcal{F}$ be a framework (V,E,$\mathbf{p}$) in general position. $\mathcal{F}$ has

- $dim\,[\mathcal{M}(\mathcal{F})]$ degrees of freedom
- $dim\,[\mathcal{M}(\mathcal{F})] - dim\,[\mathcal{R}(\mathcal{F})]$ internal degrees of freedom.

Adding a single edge to a framework adds a single linear equation to the list of linear equations defining $\mathcal{M}(\mathcal{F})$ and can reduce the dimension of $\mathcal{M}(\mathcal{F})$ by at most one. An edge is an *implied edge* if the linear equation associated with the edge is a linear combination of the linear equations associated with the other edges of $\mathcal{F}$.

Theorem 5 *Let (V,E,**p**) be a framework embedded in the plane with $|V| \geq 2$ and in general position. If $|E| < 2\,|V| - 3$, then the framework is not infinitesimally rigid.*

Proof. Let $\mathcal{F}$ = (V,E,$\mathbf{p}$) be a framework embedded in the plane in general position where E = $\left\{e_1, ..., e_{|E|}\right\}$ and let $E_i = \{e_1, e_2, ..., e_i\}$. Starting with

$$\mathcal{M}(V, E_0, \mathbf{p}) = \mathcal{R}^{2|V|},$$

the entire space, and adding edges one at a time, each edge of the framework *may* reduce the dimension of $\mathcal{M}(V, E_{i-1}, \mathbf{p})$ by at most one. Thus,

$$dim\,[\mathcal{M}(V, E_i, \mathbf{p})] \geq 2n - i$$

and

$$dim\left[\mathcal{M}(V,E,\mathbf{p})\right] \geq 2\left|V\right| - \left|E\right|.$$

So $\mathcal{F}$ can be infinitesimally rigid only if $3 \geq 2\left|V\right| - \left|E\right|$, that is, only if $\left|E\right| \geq 2\left|V\right| - 3$.

2.10 Rigidity Matrix

Let F be a framework (V,E,$\mathbf{p}$) and let $\mathbf{p}_i$ denote the vector $\mathbf{p}(i)$, so $\mathbf{p}$ maps V into $\mathbb{R}^2$, identifying $\mathbf{p}$ with a $2n$-tuple of real numbers:

$$(p_{11}, p_{21}, p_{12}, p_{22}, ..., p_{1n}, p_{2n}),$$

where (p_{1i}, p_{2i}) are the coordinates of $\mathbf{p}_i$. An edge length is a constant fixed by a quadratic equation. This implies that there are $|E|$ quadratic equations. The quadratic function is $\varphi : \mathbb{R}^{2n} \rightarrow \mathbb{R}^{|E|}$ defined by $\varphi\left(\mathbf{p}\right)_{ij} = (\mathbf{p}_i - \mathbf{p}_j)^2$. The coordinates of $\mathbb{R}^{|E|}$ are indexed by the pairs ij in lexicographical order. φ is continuously differentiable and the *Rigidity Matrix for the embedding* $\mathbf{p}$, R($\mathbf{p}$), is defined by $\varphi'(\mathbf{p}) = 2\mathrm{R}(\mathbf{p})$. R($\mathbf{p}$) is an $|E|$ by $2n$ matrix whose entries are functions for the coordinates of $\mathbf{p}$ as a point in $\mathbb{R}^{2n}$.

An example shows the rigidity matrix, where n = 4 and m = 2 and every possible edge is represented.

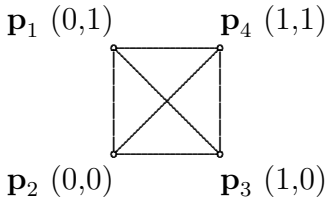


Figure 2.1: Framework with all possible edges and labeled embeddings

$$\mathbf{p} = (\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4) = (p_{11}, p_{12}, p_{21}, p_{22}, p_{31}, p_{32}, p_{41}, p_{42});$$

$$\begin{bmatrix} p_{11}-p_{21} & p_{12}-p_{22} & p_{21}-p_{11} & p_{22}-p_{12} & 0 & 0 & 0 & 0 \\ p_{11}-p_{31} & p_{12}-p_{32} & 0 & 0 & p_{31}-p_{11} & p_{32}-p_{12} & 0 & 0 \\ p_{11}-p_{41} & p_{12}-p_{42} & 0 & 0 & 0 & 0 & p_{41}-p_{11} & p_{42}-p_{12} \\ 0 & 0 & p_{21}-p_{31} & p_{22}-p_{32} & p_{31}-p_{21} & p_{32}-p_{22} & 0 & 0 \\ 0 & 0 & p_{21}-p_{41} & p_{22}-p_{42} & 0 & 0 & p_{41}-p_{21} & p_{42}-p_{22} \\ 0 & 0 & 0 & 0 & p_{31}-p_{41} & p_{32}-p_{42} & p_{41}-p_{31} & p_{42}-p_{32} \end{bmatrix}$$

Solving the equation $\mathrm{R}(\mathbf{p}) \cdot \vec{v} = 0$, the vector $\vec{v}$ is the velocity vector which gives the initial velocities for each edge. If the solution space only includes three solutions then the framework is rigid since any rigid framework has three degrees of freedom.

Now fix an edge of the framework in the plane. This means fixing zero values for elements in the $\vec{v}$. The resulting null space will have no solution if the framework is rigid. If the framework is not rigid then there will exist a solution and will require the introduction of a brace or a relocation of a brace. A brace will decrease the solution space by one if it is not overbracing an element of the framework.

Chapter 3

A Specific Framework

3.1 Rigid Construction

Given a rigid framework we can attach a new vertex by two edges to obtain a bigger rigid framework. We formulate this as a theorem for future reference. The proof can be found in Graver [2].

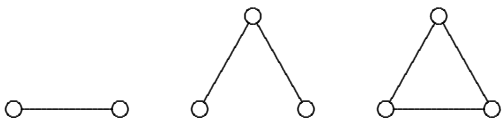


Figure 3.1: Rigid Construction

Figure 3.1 shows the statement.

Theorem 6 *Attaching two edges with a common vertex to a rigid framework results in another rigid framework.*

3.2 The Unit Grid

An *mxn Unit Grid* is a framework (V,E,**p**) embedded in the plane.

$$\mathrm{V} = \left\{a_{(0,0)}, a_{(0,1)}, ..., a_{(0,n)}, a_{(1,0)}, a_{(1,1)}, ..., a_{(1,n)}, ..., a_{(m,0)}, ..., a_{(m,n)}\right\}$$

$$E = \left\{(a_{(i,j)}, a_{(i+1,j)}), (a_{(i,j)}, a_{(i,j+1)})\right\} \text{ for } 0 \leq i \leq m \text{ and } 0 \leq j \leq n.$$

$$\mathbf{p} : \mathbf{p}(a_{(0,0)}) = (0, 0)$$

Figure 3.2 shows a 1 x 5 unit grid. *mxn* Unit Grids are not rigid. Figure 3.3

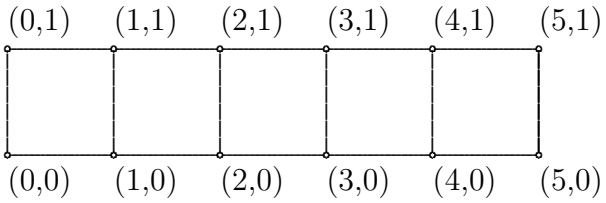


Figure 3.2: 1 x 5 Unit Grid

demonstrates a 1x1 Unit Grid, also called a *unit square*. The square framework

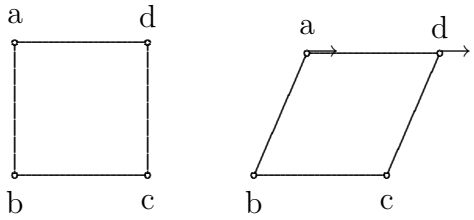


Figure 3.3: Unit Square and Deformed Unit Square

admits a deformation since the distance between the vertices a and c differs in the two embeddings. This is true for vertices b and d but not any vertices connected by an edge. Since this framework admits a deformation it is **not** rigid. This now presents an interesting question, How can this framework be made rigid? Adding an edge connecting a pair of the vertices where the deformation occurs will make the unit grid rigid, this specific edge will be named a *brace*. A *Brace* is an edge of length $\sqrt{2}$ that joins opposite vertices of a unit square framework thus making the unit square into a rigid unit square.

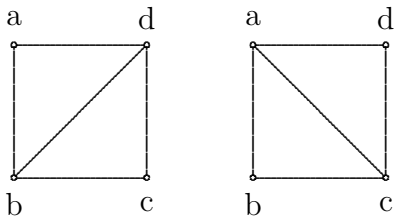


Figure 3.4: Two Rigid Unit Squares

Notice that a brace connecting vertices A and C or a brace connecting vertices B and D makes the unit square rigid just the same.

A *cell* (i, j) is the unit square composed of edges,

$$(a_{(i,j)}, a_{(i,j-1)}), (a_{(i,j-1)}, a_{(i-1,j-1)}), (a_{(i-1,j-1)}, a_{(i-1,j)}), (a_{(i-1,j)}, a_{(i,j)}).$$

where $1 \leq i \leq n$ and $1 \leq j \leq m$. A *row* i consists of cells (i, j) for $j = 1, ...m$. A *column* j consitsts of all cells (i, j) for $i = 1, ..., n$.

3.3 Brace Graph

The *brace graph* contains a vertex for each row and each column of the unit grid. The vertices will encode the bracing of the unit grid as follows: If the cell in row r_i and column c_j is braced, the vertices of the brace graph labeled r_i and c_j are joined by an edge as illustrated by Figure 3.5 and Figure 3.6.

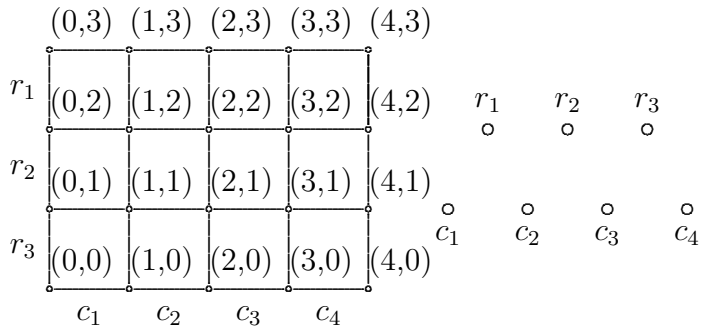


Figure 3.5: 3 x 4 Unit Grid and Associated Bipartite Graph

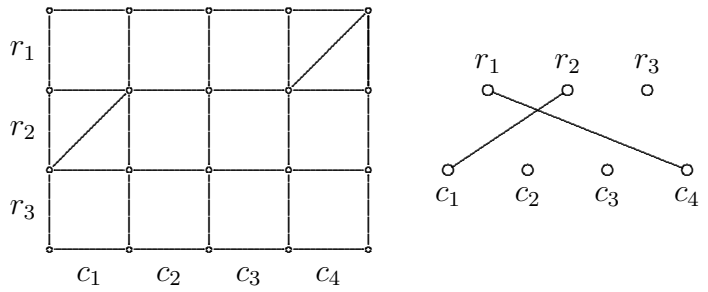


Figure 3.6: 3 x 4 Unit Grid with 2 braces and Associated Bipartite Graph

Chapter 4

Rigidifying the Unit Grid

4.1 Number of Braces

If given a unit grid constructed without braces, the unit grid will not be rigid. If each unit square is made rigid, this intuitively implies that the unit grid must be rigid. The 1 x 5 unit grid with 5 braces is now rigid but, is it over braced? i.e.

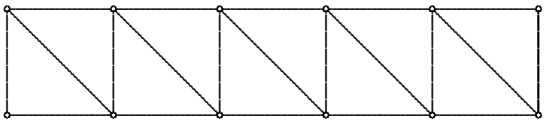


Figure 4.1: 1 x 5 Unit Grid, Overbraced?

Can a brace be removed while having the framework remain rigid? Figure 4.2 shows

Figure 4.2: Removed Brace and the Deformation

the unit grid now admits a deformation which will occur no matter which brace

is removed. The 1x5 unit grid requires that all squares be braced in order for the whole framework to be rigid. Extending this to the 1 x k unit grid case requires k braces to make the framework rigid. Let us compute how many braces are necessary to brace an mxn grid using Theorem 5. Let r denote the number edges for a rigid framework and b denote the number of braces in the mxn unit grid.

$$\begin{aligned}
|V| &= (m+1)(n+1) \\
|E| &= 2mn+m+n \\
r &= 2|V|-3 \\
r &= b+|E| \\
\rightarrow b &= r-|E| \\
&= (2|V|-3)-(2mn+m+n) \\
&= 2(m+1)(n+1)-3-(2mn+m+n) \\
&= \mathbf{m+n+1} \textit{Number of Braces for a Rigid Unit Grid} \qquad (4.1)
\end{aligned}$$

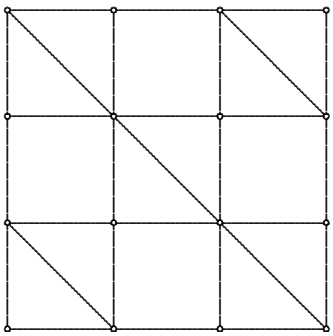


Figure 4.3: 3 x 3 Unit Grid with 5 Braces...

Figure 4.3 shows a 3x3 unit grid which by Equation 4.1 requires five braces to be rigid. As shown, there are five braces but....Figure 4.4 shows otherwise. The picture

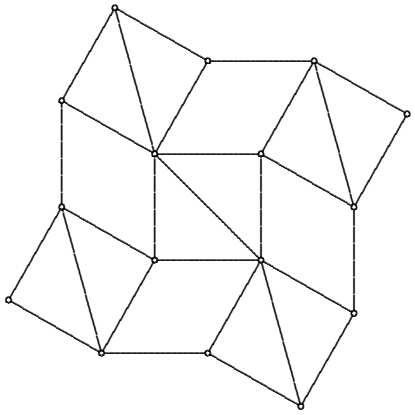


Figure 4.4: 3 x 3 Unit Grid with 5 Braces...yet not rigid!

shows a deformation, which by the definition of rigidity shows that this framework is not rigid. The condition of requiring a certain number of braces is not sufficient, the braces must also be properly placed.

4.2 Making the Unit Grid Rigid

Theorem 7 *A braced grid will be rigid if and only if its associated brace graph is connected.*

The proof is as given by Graver [2]page 52,

Proof. Consider and mxn unit grid with usual labeling of its rows and columns. Referring to an undeformed drawing of it, we will call the vertical edges in the ith row the edges of row i. Similarly, the horizontal edges in column j will be called the edges of column j. The first observation in this proof is the fact that, in any deformation, each cell is a parallelogram. It follows from this observation that, no matter how the grid is deformed, the edges of row i remain parallel to one another. Similarly, the edges of column j remain parallel to one another. And this is true for each row index i and each column index j. This observation is illustrated below,

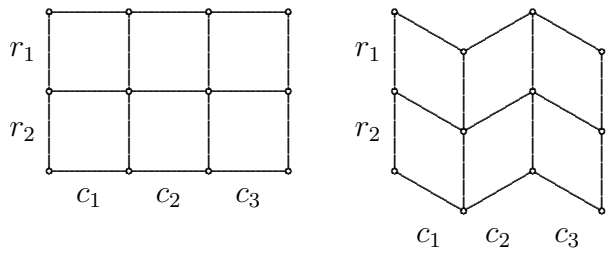


Figure 4.5: 2 x 3 Unit Grid and Deformation

Suppose that there is a brace in the unit square constructed of edges *(i,j)*, *(i+1,j)*, *(i,j+1)*,*(i+1,j+1)* of the unit grid, that is, an edge from r_i to c_j in the associated brace graph. Since, in any deformation of the braced grid, the edges of row i that bound the braced cell are perpendicular to the edges of column j that bound the braced cell, we conclude that all the edges of row i are perpendicular to all the edges of column j. Suppose that braces have been added so that there is a path in the associated bipartite graph from r_i to c_j.

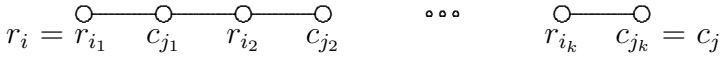


Figure 4.6: Path from r_i to c_j

Since the edges of row r_{i_1} and the edges of row r_{i_2} are perpendicular to the edges of column c_{i_1}, the edges of row r_{i_1} are parallel to the rods of row r_{i_2}. Similarly, the edges of column c_{i_1} are parallel to the edges of column c_{i_2}. Inductively,

- all the edges in all the rows of the path are parallel to one another;
- all the edges in all the columns of the path are parallel to one another; and
- all the edges in all the rows of the path a perpendicular to all the edges in all the columns of the path.

In particular, the row edges bounding the i,jth cell are perpendicular to the column edges bounding the i,jth cell. In short, if there is a path from r_i to c_j in the associated brace graph, the cell in the ith row and jth column must remain square under all motions of the braced grid. Thus, if the associated graph is connected, then there is a path from each r_i to each c_j and each cell must remain square under all motions; that is, the braced grid has no deformations. To prove the converse, assume that G is an m x n braced unit grid whose associated brace graph is not connected, show that G admits a deformation. Let A denote the set of vertices of the component of the associated brace graph that contains r_1, and let B be the set of the remaining vertices. Hence, if e is an edge either both of its endpoints are in A or both are in B. Equivalently, if the i,jth cell is braced, then either r_i and c_j are both in A or they are both in B. Notice that the edges along the top and left side of an unbraced grid may be independently reoriented. Using this observation a deformation of the braced grid is constructed. Let a denote the measure of some small angle and adjust the edges along the left side and top of the unbraced grid as follows:

- If $r_i \in$ A, the corresponding edge is vertical;
- If $c_j \in$ A, the corresponding edge is horizontal.
- If $r_i \in$ B, the corresponding edge makes a counterclockwise angle of measure a with the vertical.
- If $c_j \in$ B, the corresponding edge makes a counterclockwise angle of measure a with the horizontal.

Figure 4.7 demonstrates this construction.

$$\text{A} = \{r_1, r_3, c_2, c_3, c_5\} \text{ B} = \{r_2, c_1, c_4\}.$$

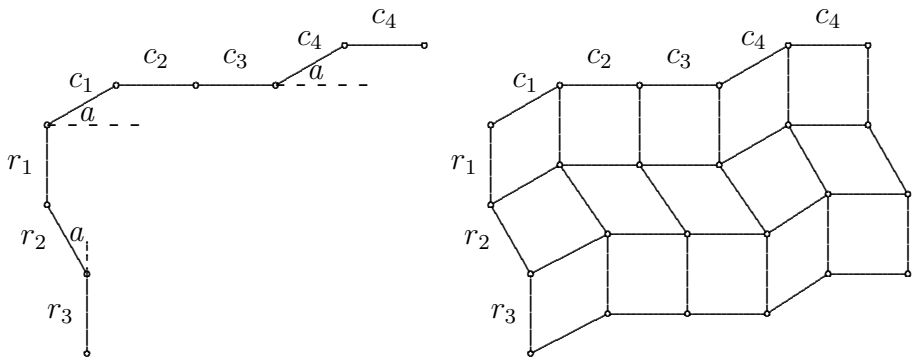


Figure 4.7: Constrution of Deformation

The figure shows that sets of the deformed grid whose row and column vertices are in the same set are square! Thus all of the square that could be braced are. Hence, the undeformed braced grid can be deformed by increasing a from 0 to some positive value.

The results allow the solution for the rigidity of a unit grid to become apparent through observing the connectivity of the associated brace graph. To show that this is true, let's revisit some of the previous examples and their brace graph.

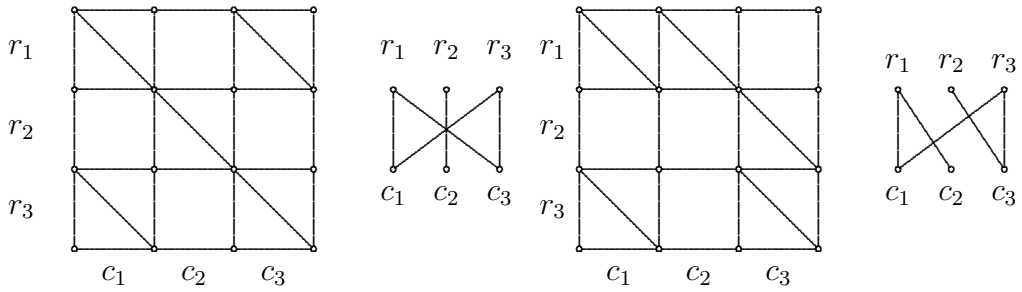


Figure 4.8: 3 x 3 Unit Grids with associated Bipartite Graphs

In Figure 4.8 the left hand figure is not rigid while the figure on the right side is rigid. Notice the accompanying brace graphs, the associated brace graph for the figure on the left side is *disconnected* while for the figure on the right the associated

graph is connected. This example shows the solution.

Chapter 5

A New Problem

5.1 Unit Grid with a Window

Given an mxn unit grid, delete a vertex $a_{(i,j)} \in V$ such that $i \neq 0, 1, n-1, n$ and $j \neq 0, 1, m-1, m$, and all the edges incident with vertex $a_{(i,j)}$. This creates a $2x2$ *window* . There must not exist a brace within cells $(i, j), (i+1, j), (i+1, j+1)$, and $(i, j+1)$. The smallest unit grid where a window can exist is a $4x4$ unit grid.

Figure 5.1 shows a 3 x 3 unit grid where the vertex deleted does not satisfy the conditions stated to create a window. The window created is $2x2$ on the border of the framework. A 3 x 3 unit grid requires at least 5 properly placed braces to be rigid, they will be placed in the only configuration allowed. Considering Figure 5.1,

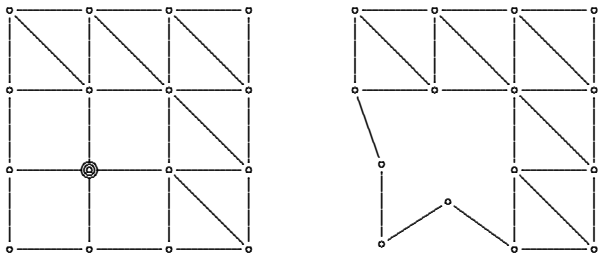


Figure 5.1: 3 x 3 Unit Grid with Vertex (1,1) Deleted

the vertex deleted must be sufficiently in the interior of the framework so that the 2 x 2 window created is surrounded by unit squares on all sides. This condition then requires that the unit grid containing the 2 x 2 window must at least be a 4 x 4 unit grid. Requiring the window to be surrounded by unit squares allows braces to be placed around the window to achieve rigidity.

5.2 Physical Model

The unit grid is a simple enough framework where a physical model will give some indication as to rigidity and the effects of a window on the dynamics of a unit grid. I built a model using wooden pieces to represent edges and joined the edges together by nails to represent the freemoving vertices. The model worked very well in giving a feeling for the framework and its rigidity, yet there were several limitations. The vertices were loose, allowing for motion not expected from the model developed thus far. The motion became compounded as more vertices were added and the framework became larger and larger. Rigidity in the physical model was defined slightly different, allowing for some movement in the model. If a unit square completely collapsed, the model was not rigid. Any motion other than that was caused by model inaccuracies.

Another issue was that as the model became larger and larger the vertices were connecting more and more edges. The edges did not remain in the plane, i.e. the surface it rested on. Edges sharing the same vertex force the model to not be in the plane. The varying heights, though small, influenced the movement of the model. Motions of the model in which vertices move out of the plane are again not predicted by the theory.

5.3 Experimentation

To determine the rigidity of a braced grid with a window we no longer use the brace graph.

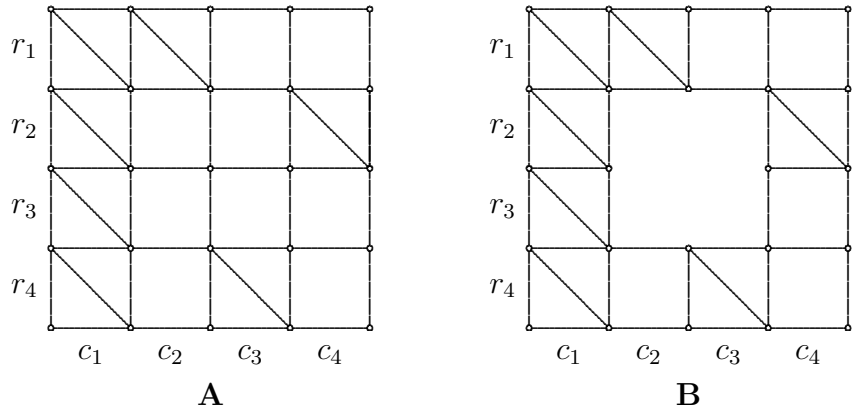


Figure 5.2: Braced 4 x 4 Unit grid with and without a window

The two unit grids are braced exactly the same. The grid on the left is rigid, as shown by the brace graph in Figure 5.3. The brace graphs are exactly the same for both figures, but the figure in the right is not rigid.

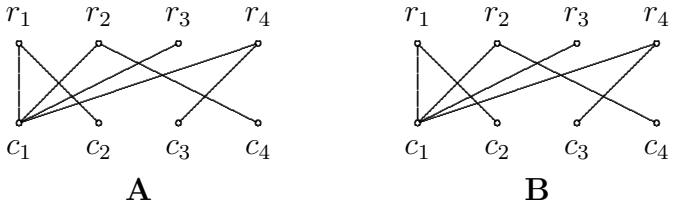


Figure 5.3: Brace Graphs for Unit Grids A and B

5.4 The Architect's Problem

The original problem presented was a 6 x 10 Unit Grid with a 2 x 2 window created by deleting the vertex at (3,2). Since the brace graph cannot be used as it was to

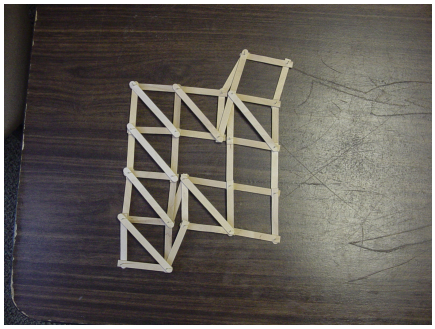


Figure 5.4: Nonrigid 4 x 4 Unit Grid with Window

indicate rigidity for the unit grid, the physical model provides a tool for understanding the rigidity of the unit grid containing a window. According to Equation 4.1, the 6 x 10 unit grid requires 15 properly braces to achieve rigidity. The 15 braces are placed in such a fashion that when the vertex at (3,2) is deleted there aren't any braces deleted. The brace graph confirms what the physical model demonstrates. Vertex (3,2) is deleted from Figure 5.5 creating the window shown in Figure 5.6.

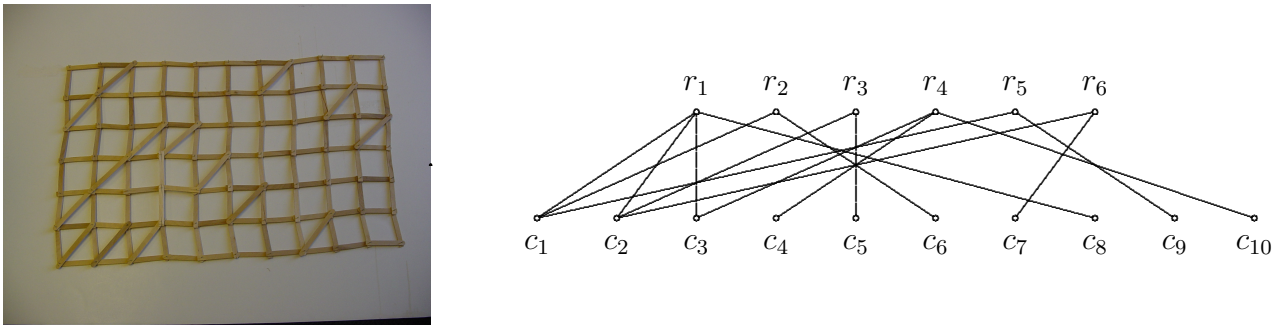


Figure 5.5: Rigid 6 x 10 Unit Grid

Notice also that the brace graphs are identical. This occurs since the edges of the brace graph represents a brace and none of the braces were deleted. The vertices of the brace graph represent the rows and columns of the unit grid and neither were deleted. Figure 5.7 shows there are areas of the unit grid that collapse. Three braces will inhibit this collapse. Using this fact, moving through the unit grid in-

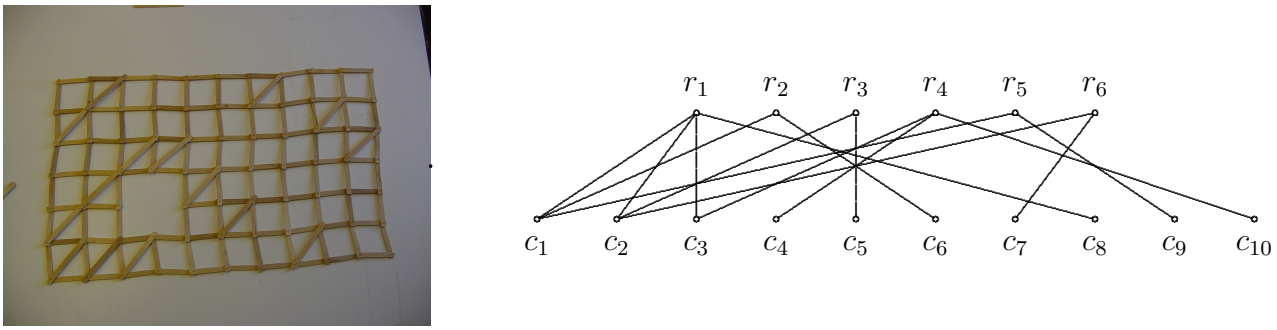


Figure 5.6: 6 x 10 Unit Grid with Window

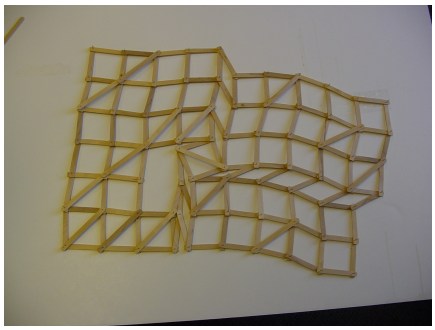


Figure 5.7: Deformation of the 6 x 10 Unit Grid with Window

hibiting areas of collapse where they occur eventually rigidifies the whole unit grid. The rigid unit grid has 18 braces. The problem of rigidifying the 6 x 10 unit grid

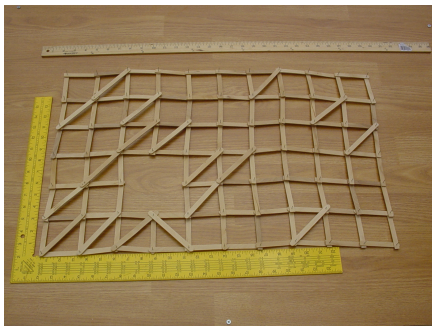


Figure 5.8: Rigidified 10 x 6 Unit Grid with Window

with a window is done, but is the placement or number of braces optimal?

5.5 Minimal Bracing

Are 18 braces in the 6 x 10 Unit Grid with a window the minimum number of braces required? Deleting the brace in the square in row 3 and column 5 does not affect the rigidity of the figure. Can more braces be deleted?

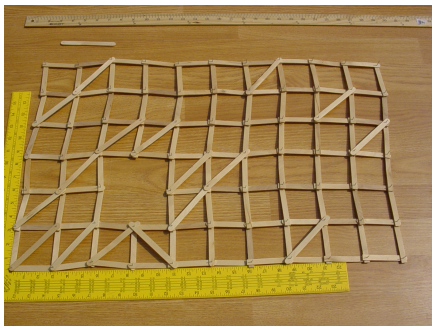


Figure 5.9: Rigidified Unit Grid containing a Window with 17 Braces

Figure 5.9 is a demonstration of a configuration for rigidity using 17 Braces.

Notice in Figure 5.10 that the brace graph is connected and the unit grid with a window is rigid. Another case demonstrates the fact that connectivity does not

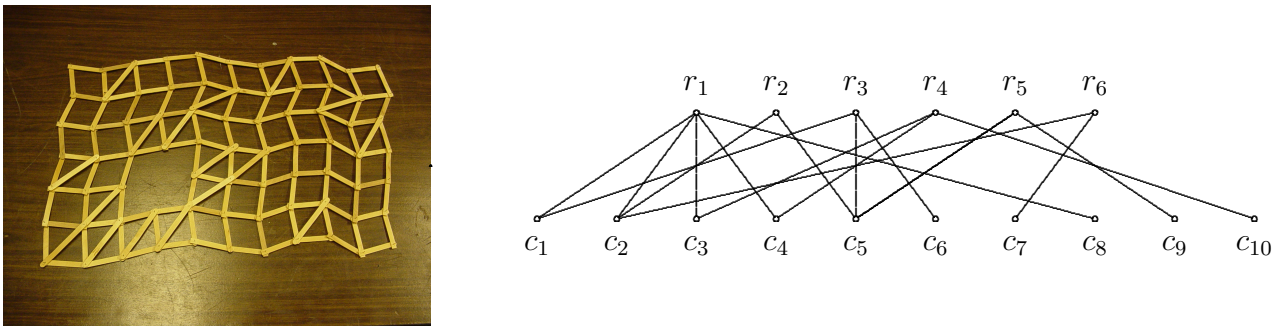


Figure 5.10: Rigidified Unit Grid containing a Window with 17 Braces

imply rigidity in the case of a unit grid with a 2 x 2 window. As the model is tested

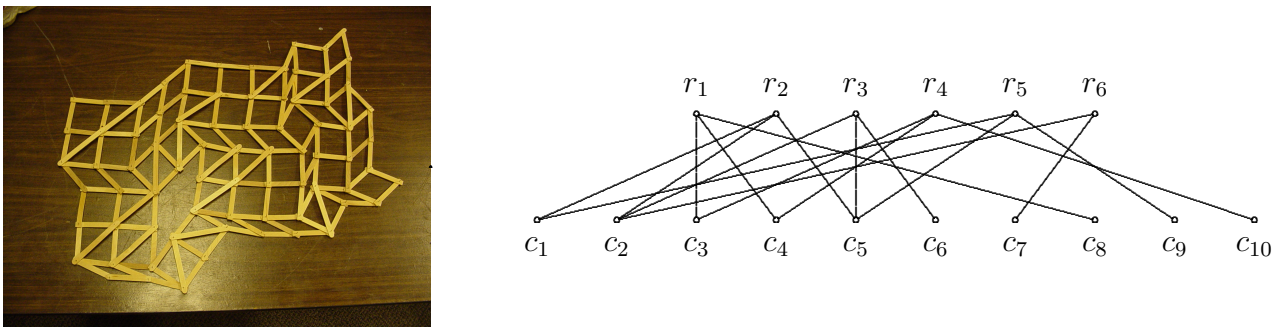


Figure 5.11: Non -Rigid Framework with 17 Braces and Connected Brace Graph

with several different configurations of bracing there are many interesting things that occur within the model. Finding the minimal bracing for this model is still the goal of subsequent trials and the results are as follows.

Figure 5.12 along with the brace graph is very interesting. The brace in row 5 column 5 was removed. First examine the brace graph, the edge connecting r_5 and c_9 is disconnected from the rest of the graph. Figure 5.12 also shows a non-rigid model notice that part of the model that is rigid. There is a rigid 4 x 8 rectangle that remains rigid in the model while a row and a column completely collapse. The subsection that is rigid has 12 braces among its rows and columns. The row and column

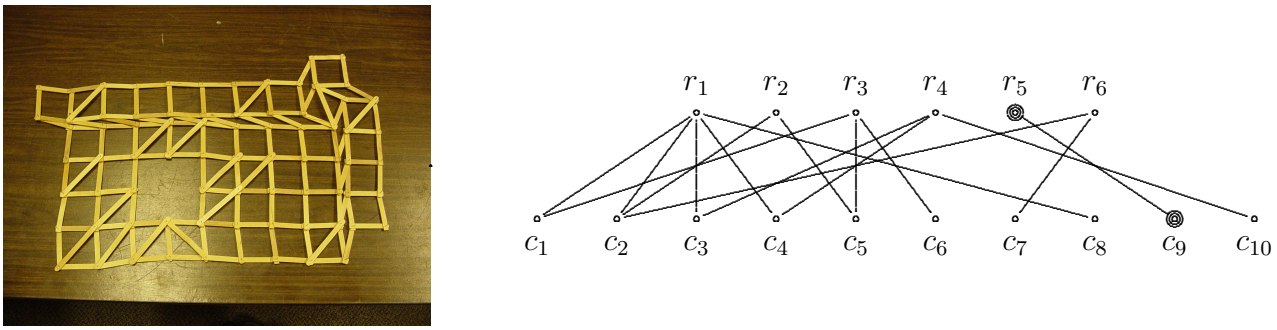


Figure 5.12: Non-Rigid Framework with 16 Braces and Disconnected Brace Graph

that comletely collapse are joined by an edge in the brace graph and disconnected from the rest. This leads to the possibility that the rigid rectangle may be rigid on its own, but looking at column 7 among the rigid rectangle shows there is no brace. Figure 5.13 shows that though the rectangle is rigid among the model it is

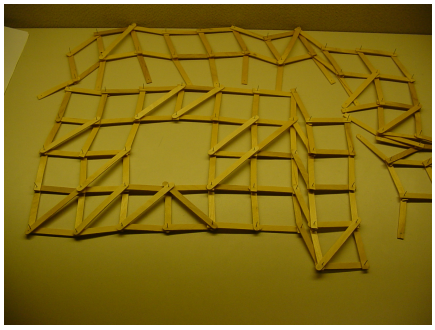
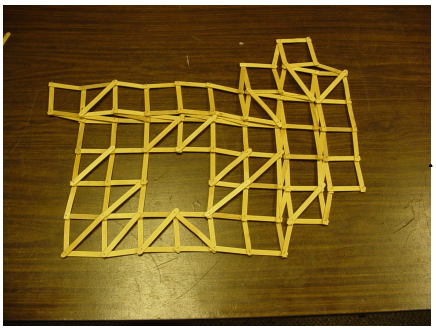


Figure 5.13: 4 x 8 SubSection of Figure 2.7

not rigid itself. A brace added to the collapsed column would create rigidity. Thus the sub-section would be rigid with 13 braces. This shows that it may be possible to create a rigid element of the framework which may be used to work from to find a minimal bracing. Notice again that there is a disconnected element of the brace graph in Figure 5.14 and that element corresponds to the part of the model that is completely collapsed. This model also has a rigid sub-section. The sub-section that occurs among this model is rigid on its own. The sub-section has 11 properly placed



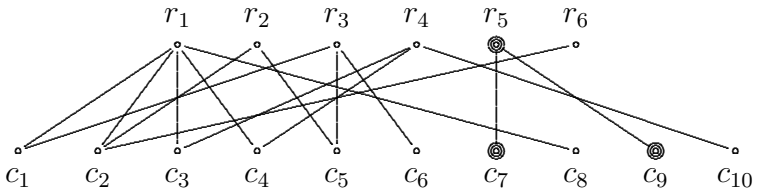


Figure 5.14: Non-Rigid Framework with 16 Braces and Disconnected Brace Graph

braces that rigidify it.

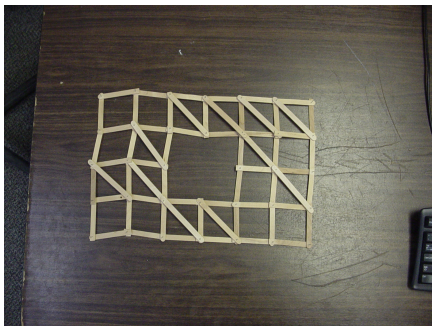


Figure 5.15: Rigid Sub-Section with 11 Braces

A rigid model now exists. The model is minimally braced since if one brace was removed it is no longer rigid. From here a larger minimally rigid model may be created. Another configuration creates an interesting model.

5.6 Minimal Framework

Let us first study a $4x4$ unit grid with a $2x2$ window in the $4x4$ unit grid case. Begin by overbracing the model with placing braces in every unit square available. The result is that every unit square surrounding the window is braced.

The removal of any brace within a unit square that shares an edge with the

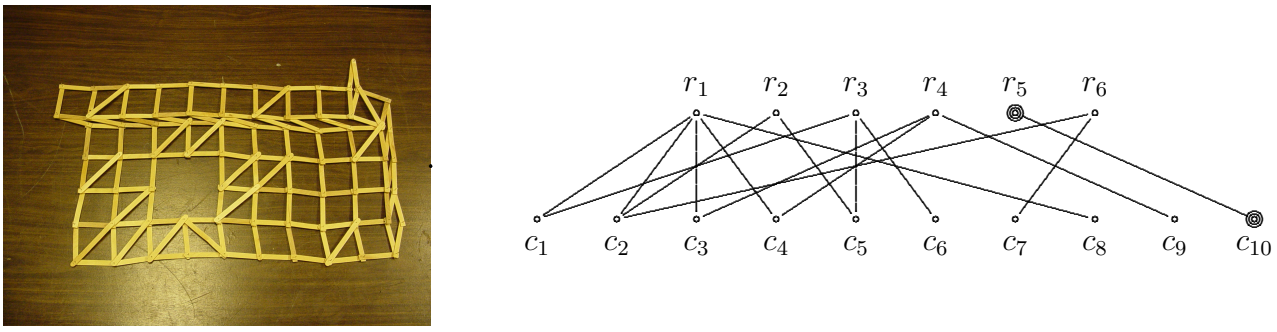


Figure 5.16: Non-Rigid Framework with 16 Braces and Disconnected Brace Graph

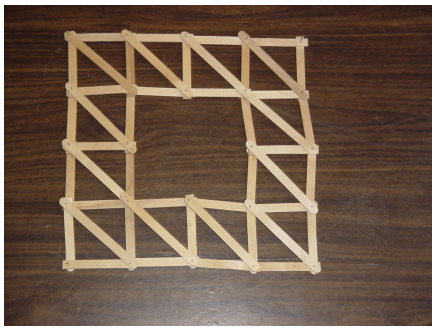


Figure 5.17: Overbraced 4 x 4 Unit Grid with Window

window results in a deformation. This implies that the unit squares sharing an edge with the window must be made rigid either by a brace or as a unit square apart of a larger rigid sub-section. The only unit squares left to remove a brace from are the corner unit squares. Removing all but one of the corner unit square braces results in rigidity, removing all of the corner unit square braces results in a deformation.

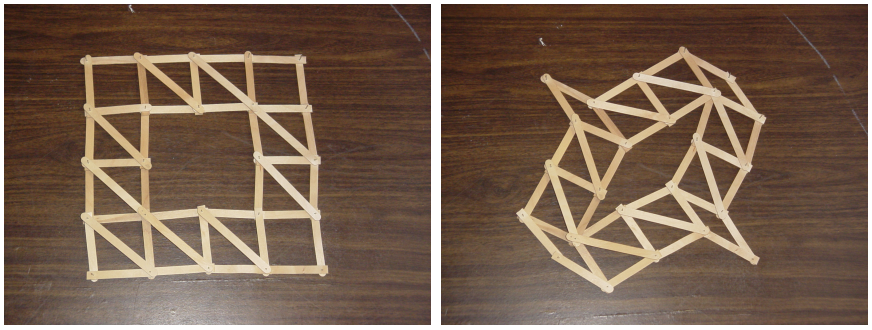


Figure 5.18: Minimally Braced Rigid 4 x 4 Framework and Nonrigid 4 x 4 Framework

Figure 5.18 shows the minimally rigid 4 x 4 unit grid with a window. The 4 x 4 unit grid with a window requires 9 braces. In Figure 5.15, the framework is a 4 x 6 unit grid with a window that requires 11 braces. The pattern that is developing is that for an h x k unit grid with a 2 x 2 window contained entirely in the interior of the framework, the framework requires at least h + k + 1 properly placed braces. The proof of this statement is constructive.

Begin with the 4 x 4 unit grid with a window contained entirely in the interior of the 4 x 4 unit grid. Place braces in every unit square that shares an edge with the window and in one corner unit square as well. This unit grid is minimally rigid since the removal of a single brace results in a deformation. The number of braces for this framework is equal to the dimensions added plus 1.

A larger unit grid requires adding subsequent rows and columns. For each row or column added there is only one brace required to maintain rigidity. Using Axiom 6, attach two edges to the rigid unit grid with a window where one edge is a brace.

The framework remains rigid. The framework is not quite a unit grid but it is a rigid framework. This framework requires only adding two edges joined by a vertex to fill out the row or column, once this is achieved another row or column may be created by beginning with the brace and the edge. The process can be continued indefinitely and the formula holds, that for and h x k unit grid containing a 2 x 2 window requires h + k + 1 properly placed braces.

This is optimal because given an hxk unit grid with a $2x2$ window,

$$|V| = (h+1)(k+1) - 1$$

$$\text{Vertical and Horizontal Edges} = (2hk + h + k) - 4$$

$$braces = h + k + 1$$

therefore

$$|E| = ((2hk + h + k) - 4) + h + k + 1$$

satisfies the degrees of freedom equation,

$$|E| = 2\,|V| - 3,$$

Chapter 6

An Algebraic Solution

6.1 Program

Numerical computation will find the null space of the rigidity matrix for the unit grid with a window. (The program is attached.) The solution space will be 3-dimensional if the unit grid with a window is rigid. To rule out isometries of the plane, we may pin an edge. An edge is pinned if its given initial velocity is zero. In the program the rigidity matrix included three rows where the elements corresponding to the coordinates of the vertex in the upper left corner and the x coordinate of the vertex below the corner were ones. This forced the first three entries of the velocity vector to be zero.

6.2 Numerical Computation Results

The program will now give a definitive answer regarding the rigidity of the figures reviewed thus far. The programs given in the appendix worked together to numerically support the findings presented. The RigidityMatrix program found the nullspace for a unit grid with an edge pinned not containing a window. The program used both

the UnitGrid and bracegraphmatrix programs. The RigidityMatrixWindow found the nullspace of a unit grid with a pinned edge containing a window using the same support programs.

Chapter 7

Open Problems

1. How must a unit grid be braced that contains more than one $2x2$ window?

    - Given a unit grid unbraced can the same ideas of bracing a unit grid with one $2x2$ window be applied to a unit grid containing several windows?

2. The brace graph is the solution for finding how the unit grid must be braced for rigidity. Can it be modified to indicate the bracing for rigidifying a unit grid containing a window or several windows?

3. Is there a way to find an optimal configuration without computing the nullspace of the rigidity matrix of the unit grid with a window?

Appendix A

Programs

A.1 Unit Grid.m Program

These are the programs that numerically supported the conclusions.

```
0001 function [V,xcoordinate,ycoordinate,deletedvertex] = Unit_Grid(rows, columns)
0002
0003 %   Creating the coordinates for a unit grid.
0004 %   The coordinates of the lower left corner are always (0,0).
0005 %   Following the initial position the user must then indicate
0006 %   the size of the unit grid they wish to work with.
0007
0008 %A = input('enter the coordinates of the lower left hand corner of the unit gri
0009 A = [0 0];
0010 % rows = input('How many rows do you want the unit grid to have?   ');
0011 % columns = input('How many columns do you want the unit grid to have?    ');
0012 rows;
0013 columns;
0014
0015 xcoordinate = input('what is the x coordinate of the deleted vertex?   ');
0016 while (xcoordinate<=1|columns-1<=xcoordinate)
0017     disp('X coordinate of deleted vertex must be greater than one and at least
0018     xcoordinate = input('what is the x coordinate of the deleted vertex?   ');
0019 end
0020
0021 ycoordinate = input('what is the y coordinate of the deleted vertex?   ');
0022 while (ycoordinate<=1|rows-1<ycoordinate)
0023     disp('Y coordinate of deleted vertex must be greater than one and at least
```

```
0024     ycoordinate = input('what is the x coordinate of the deleted vertex?   ');
0025 end
0026
0027 % ycoordinate = input('What is the y coordinate of the deleted vertex?   ');
0028 % while (ycoordinate<=1|rows-1<=ycoordinate)
0029 %     disp('Y coordinate of deleted vertex must be greater than one and at leas
0030 %     ycoordinate = input('what is the y coordinate of the deleted vertex?   ')
0031 % end
0032
0033
0034
0035 h = rows + 1; %   number of rows in matrix
0036 k = columns + 1; %   number of columns in matrix
0037
0038 V= zeros(h,2*k);    %   This is going to be the matrix of the embedding
0039                     %   for the unit grid.
0040
0041 Ycoords = zeros(h,1);   %   This is going to be the
0042                         %   matrix of the y coordinates
0043                         %   of the ordered pair for each vertex
0044
0045 Xcoords = zeros(h,k);   %   this matrix will hold the x coordinates of the
0046                         %   ordered pair for each vertex.
0047
0048 for m = 1:h %   this 'for' loop creates the y coordinate
0049             %   matrix
0050     Ycoords(m,1) = h-m; %   this gives the y coordinate
0051                         %   of the top row of the
0052                         %   unit grid
0053
0054     m = m + 1;
0055 end
0056
0057
0058 for n = 1:k %   This loop inserts the Ycoords matrix
0059             %   the even numbered columns of unit grid
0060             %   embedding matrix.
0061
0062     V(:,2*n) = Ycoords; %   Here the Ycoords matrix becomes
0063                         %   the column of V
0064     n = n+1;
0065 end
0066
```

```
0067
0068
0069 for i = 1:k % Creating the matrix of X coordinates
0070     Xcoords(:,i) = A(1,1) + (i-1);  % making the columns of the Xcoords matrix
0071
0072     i  = i + 1;
0073 end
0074
0075 for j = 1:k % Inserting the x coordinates from Xcoords into
0076             % the matrix V
0077     V(:,(2*j) - 1) = Xcoords(:,j);
0078
0079     j = j +1;
0080 end
0081 V(rows- ycoordinate + 1, 2*xcoordinate + 1) = 110*rows;
0082 V(rows - ycoordinate + 1, 2*xcoordinate + 2) = 115*rows;
0083
0084 vertexcounter = 0;
0085 deletedcolumn = 0;
0086 endsearch = 0;
0087 if endsearch == 0
0088     for j = 1:size(V,2)
0089         if endsearch == 0
0090             for i = 1:size(V,1)
0091                 if endsearch ==0
0092                     vertexcounter = vertexcounter + 1;
0093                     if V(i,j) == (110*rows)
0094                         endsearch = 1;
0095                         deletedcolumn = j;
0096                     end
0097                 end
0098             end
0099         end
0100     end
0101 end
0102
0103
0104 edgedifference = (rows+1)*((deletedcolumn - 1)/2);
0105 deletedvertex = vertexcounter - edgedifference;
0106
0107 %disp('THE FOLLOWING IS A MATRIX OF THE ORDERED PAIRS OF THE UNIT GRID SPECIFIE
0108 %disp(V);
0109
```

```
0110
```

A.2 Brace Graph Matrix.m Progam

```
function [M,r,c,Incident]=brace_graph_matrix
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Setting up to Solve R*v = 0                                          %
%---------------------------------------------                           %
%                                                                        %
%   INPUT                                                                %
%   -----                                                                %
%        1.) the rows and columns of the unit grid                       %
%        2.) the locations of the braces                                 %
%               -the braces are asked from the top left corner down      %
%                and then proceeding to the next column down through     %
%                exhausting the squares of the unit grid.                %
%   OUPUT                                                                %
%   -----                                                                %
%       M   :    The matrix of the bracing for the given unit grid, where%
%                a one denotes a brace in the row and column it lies and %
%                a zero denotes there isn't a brace in the row and column%
%                it lies.                                                %
%                    -This output can be inputed using                   %
%                     brace_graph_matrix.m or can be used from a        %
%                     workspace.                                         %
%       r :      the number of rows of the unit grid.                    %
%       c :      the number of columns of the unit grid.                 %
%                                                                        %
%                                                                        %
%   DESCRIPTION                                                          %
%   -----------                                                          %
%       This code takes the outputs of brace_graph_matrix.m and         %
%       Unit_Grid.m and finds the Rigidity Matrix and null space for    %
%       the given unit grid and its bracing.                             %
%       It starts by finding the rigidity matrix for the Vertical,       %
%       Horizontal, and Brace edges then combining them for the full     %
%       Rididity Matrix.  The Rigidity Matrix also has three rows on the%
%       bottom containing a one to represent the coordinate that will be%
%       pinned.  This decrease the null space by three dimensions since %
%       the 3 rigid motions can no longer occur with a pinned edge.      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

r = input('How many rows?');
c = input('How many columns?');

```

```
0042 X = zeros(r,1);
0043 M = zeros(r,c);
0044
0045 for j = 1:c              % inputting the location of the braces.
0046     for i = 1:r
0047         X(i,1) = input('Is there a brace in row i and column 1?; 1=yes, 0=no
0048         i = i + 1;
0049     end
0050     M(:,j) = X(:,1);
0051     j = j + 1;
0052 end
0053 M;
0054
0055 Incident = zeros(sum(sum(M)),r+c);
0056 IncidentRows = zeros(sum(sum(M)),r);
0057 IncidentColumns = zeros(sum(sum(M)),c);
0058 counter = 0;
0059 for j = 1:c
0060     for i = 1:r
0061     if M(i,j)>0
0062         counter = counter + 1;
0063         IncidentRows(counter,r-(i-1)) = 1;
0064         IncidentColumns(counter,j) = 1;
0065     end
0066     end
0067 end
0068 Incident = [IncidentRows IncidentColumns];
0069
```

A.3 Rigidity Matrix.m Program

```
function [RigidityMatrix,NullSpace,VertexPinning] = Rigidity_Matrix(M)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Setting up to Solve R*v = 0
%---------------------------------------------                          %
%                                                                       %
%   INPUT                                                               %
%   -----                                                               %
%       M   :    The matrix of the bracing for the given unit grid, where%
%                a one denotes a brace in the row and column it lies and %
%                a zero denotes there isn't a brace in the row and column%
%                it lies.                                               %
%                    -This output can be inputed using                  %
%                     brace_graph_matrix.m or can be used from a       %
%                     workspace.                                        %
%                                                                       %
%   OUPUT                                                               %
%   -----                                                               %
%       RigidityMatrix :    This is the rigidity matrix of the given    %
%                           unit grid and its bracing.                  %
%       Nullspace :         The solution to R*v = 0.                    %
%       VertexPinning :     Gives the location of the vertices pinned.  %
%                                                                       %
%                                                                       %
%   DESCRIPTION                                                         %
%   -----------                                                         %
%       This code takes the outputs of brace_graph_matrix.m and        %
%       Unit_Grid.m and finds the Rigidity Matrix and null space for    %
%       the given unit grid and its bracing.                            %
%       It starts by finding the rigidity matrix for the Vertical,      %
%       Horizontal, and Brace edges then combining them for the full    %
%       Rididity Matrix.  The Rigidity Matrix also has three rows on the%
%       bottom containing a one to represent the coorinate that will be %
%       pinned.  This decrease the null space by three dimensions since %
%       the 3 rigid motions can no longer occur with a pinned edge.     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if (nargin==0)
  [M,r,c]=brace_graph_matrix;
else
    r = size(M,1);
    c = size(M,2);
```

```
0042 end
0043 %[V,xcoordinate,ycoordinate] = Unit_Grid(r,c);
0044 xcoordinate = 0;
0045 ycoordinate = 0;
0046 vertices = (r+1)*(c+1);     %number of vertices in Unit Grid
0047 NumberVerticalEdges = (c+1)*r;    % number of vertical edges
0048 Vertical = zeros(NumberVerticalEdges,2*vertices); %Rigidity Matrix of vertical e
0049 VerticalSubMatrix = zeros(r,2*(r+1));
0050
0051 i = 1;
0052
0053 for i = 1:r
0054     VerticalSubMatrix(i,2*i-1) = 0;     %   sets up sub matrix for rigidity
0055     VerticalSubMatrix(i,2*i) = 1;       %   matrix of vertical edges
0056     VerticalSubMatrix(i,2*i+1) = 0;
0057     VerticalSubMatrix(i,2*i+2) = -1;
0058     i= i+1;
0059 end
0060 i = 1;
0061 for i = 1:(c+1)
0062     Vertical(r*(i-1) +1:r*i,(2*(r+1))*(i-1) + 1:2*(r+1)*i) = VerticalSubMatrix;
0063 end
0064
0065 VerticalEdges = ones(r,c+1);        % sets up to delete any vertical edges
0066 if xcoordinate >0                   % if a vertex is deleted
0067     VerticalEdges(r - (ycoordinate-1),xcoordinate + 1 )=0;
0068     VerticalEdges(r - ycoordinate, xcoordinate + 1) = 0;
0069 end
0070
0071 i = 1;
0072 j = 1;
0073 counter = 1;
0074 for j = 1:c
0075     for i = 1:r
0076         if VerticalEdges(i,j) < 1
0077             Vertical(counter,:)=0;
0078             counter = counter + 1;
0079         else
0080             counter = counter + 1;
0081         end
0082         i = i + 1;
0083     end
0084     j = j + 1;
```

```
0085 end
0086
0087 NumberHorizontalEdges = c*(r+1);
0088 Horizontal = zeros(NumberHorizontalEdges,2*vertices);
0089
0090 i = 1;
0091 j = 1;
0092 for i = 1:(r+1)*c                          % sets up Rigidity matrix for
0093     Horizontal(i,2*i-1) = -1;              % horizontal edges.
0094     Horizontal(i,2*i) = 0;
0095     Horizontal(i,2*r + 2*i + 1) = 1;
0096     Horizontal(i,2*r + 2*i + 2) = 0;
0097     i = i+1;
0098 end
0099
0100 HorizontalEdges = ones(r+1,c);             % sets up to delete any horizontal
0101 if xcoordinate>0                           % edges if necessary
0102     HorizontalEdges(r-ycoordinate+1,xcoordinate) = 0;
0103     HorizontalEdges(r-ycoordinate+1,xcoordinate + 1) = 0;
0104 end
0105
0106 counter = 1;
0107 i = 1;
0108 j = 1;
0109 for j = 1:c
0110     for i = 1:r+1
0111         if HorizontalEdges(i,j) < 1
0112             Horizontal(counter,:)=0;
0113             counter = counter + 1;
0114         else
0115             counter = counter + 1;
0116         end
0117         i = i + 1;
0118     end
0119     j = j + 1;
0120 end
0121
0122
0123 BraceEdges = sum(sum(M));   %number of braces in Unit Grid
0124 BraceSubMatrixA = zeros(r,2*r);
0125 BraceSubMatrixB = zeros(r,2*r);
0126 i = 1;
0127 for i = 1:r
```

```
0128     BraceSubMatrixA(i,2*i-1) = -1;        % sets up sub matrix for rigidity
0129     BraceSubMatrixA(i,2*i) = 1;           % matrix of all possible brace
0130     BraceSubMatrixB(i,2*i-1) = 1;         % edges
0131     BraceSubMatrixB(i,2*i) = -1;
0132     %Brace(i,2*r + 2*i + 3) = 1;
0133     %Brace(i,2*r + 2*i + 4) = -1;
0134     i = i+1;
0135 end
0136
0137 Brace = zeros(r*c,2*vertices);
0138 i = 1;
0139
0140 for i = 1:c
0141     Brace(r*(i-1) + 1:r*i,(2*r + 2)*(i-1) + 1:(2*r + 2)*(i-1) + 2*r) = BraceSubl
0142     Brace(r*(i-1) + 1:r*i,(2*r + 5) + (i-1)*(2*r+2):4*r + 4 + (i-1)*(2*r+2)) = I
0143     i = i + 1;
0144 end
0145
0146 counter = 1;        % finds and deletes brace edges that are not present
0147 i = 1;              % in the unit grid.
0148 j = 1;
0149 for j = 1:c
0150     for i = 1:r
0151         if M(i,j) < 1
0152             Brace(counter,:) = 0;
0153             counter = counter + 1;
0154         else
0155             counter = counter + 1;
0156         end
0157         i = i + 1;
0158     end
0159     j = j + 1;
0160 end
0161
0162 BraceCheck = zeros(1,2*vertices);
0163 DeletedRow = zeros(1,2*vertices);
0164 counter = 0;
0165 i = 1;
0166 while i <= size(Brace,1)
0167     BraceCheck(1,:) = Brace(i,:);
0168     if BraceCheck == DeletedRow;
0169        Brace(i,:) = [];
0170        i = i - 1;
```

```
0171     end
0172     i = i + 1;
0173 end
0174
0175 RigidityMatrix = [Vertical; Horizontal; Brace];
0176 ExtraZeros = zeros((2*vertices) - size(RigidityMatrix,1)-4,2*vertices);
0177 VertexPinning = zeros(3,2*vertices);
0178 VertexPinning(1,1) = 1;     % adds rows to pin an edge the upper left
0179 VertexPinning(2,2) = 1;     % vertical edge.
0180 VertexPinning(3,3) = 1;
0181 %VertexPinning(4,4) = 1;
0182 RigidityMatrix = [RigidityMatrix; ExtraZeros;VertexPinning];
0183
0184 NullSpace = null(RigidityMatrix,'r');
0185
```

A.4 Rigidity Matrix Window.m Program

```
0001 function [NullSpace,M] = Rigidity_MatrixWindow(M)
0002 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0003 %   Setting up to Solve R*v = 0                                                %
0004 %-------------------------------------------                                  %
0005 %                                                                            %
0006 %   INPUT                                                                    %
0007 %   -----                                                                    %
0008 %       M  :    The matrix of the bracing for the given unit grid, where%
0009 %               a one denotes a brace in the row and column it lies and %
0010 %               a zero denotes there isn't a brace in the row and column%
0011 %               it lies.                                                     %
0012 %                   -This output can be inputed using                        %
0013 %                    brace_graph_matrix.m or can be used from a             %
0014 %                    workspace.                                              %
0015 %                                                                            %
0016 %   OUPUT                                                                    %
0017 %   -----                                                                    %
0018 %       Nullspace :         The solution to R*v = 0.                        %
0019 %       M :                 The Bracing of the Unit Grid                    %
0020 %                                                                            %
0021 %                                                                            %
0022 %   DESCRIPTION                                                              %
0023 %   -----------                                                              %
0024 %       This code takes the outputs of brace_graph_matrix.m and             %
0025 %       Unit_Grid.m and finds the Rigidity Matrix and null space for        %
0026 %       the given unit grid and its bracing.                                 %
0027 %       It starts by finding the rigidity matrix for the Vertical,           %
0028 %       Horizontal, and Brace edges then combining them for the full        %
0029 %       Ridідity Matrix.  The Rigidity Matrix also has three rows on the%
0030 %       bottom containing a one to represent the coordinate that will be%
0031 %       pinned.  This decreases the null space by three dimensions since %
0032 %       the 3 rigid motions can no longer occur with a pinned edge.      %
0033 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
0034
0035 if (nargin==0)
0036   [M,r,c]=brace_graph_matrix;
0037 else
0038     r = size(M,1);
0039     c = size(M,2);
0040 end
0041 [V,xcoordinate,ycoordinate,deletedvertex] = Unit_Grid(r,c);
```

```
0042 %xcoordinate = 0;
0043 %ycoordinate = 0;
0044 vertices = (r+1)*(c+1);     %number of vertices in Unit Grid
0045 NumberVerticalEdges = (c+1)*r;    % number of vertical edges
0046 Vertical = zeros(NumberVerticalEdges,2*vertices); % Rigidity Matrix of
0047 VerticalSubMatrix = zeros(r,2*(r+1));             % vertical edges
0048
0049 i = 1;
0050
0051 for i = 1:r
0052     VerticalSubMatrix(i,2*i-1) = 0;     % sets up the submatrix that makes
0053     VerticalSubMatrix(i,2*i) = 1;       % up the rigidity matrix for the
0054     VerticalSubMatrix(i,2*i+1) = 0;     % vertical edges of a unit grid
0055     VerticalSubMatrix(i,2*i+2) = -1;
0056     i= i+1;
0057 end
0058 i = 1;
0059 for i = 1:(c+1)
0060     Vertical(r*(i-1) +1:r*i,(2*(r+1))*(i-1) + 1:2*(r+1)*i)...
0061         = VerticalSubMatrix;
0062 end
0063
0064 VerticalEdges = ones(r,c+1);
0065 if xcoordinate >0
0066     VerticalEdges(r - (ycoordinate-1),xcoordinate + 1 )=0;
0067     VerticalEdges(r - ycoordinate, xcoordinate + 1) = 0;
0068 end
0069
0070 i = 1;                  % makes the rows zeors for the edges associated
0071                         % with the deleted vertex
0072 j = 1;
0073 counter = 1;
0074 for j = 1:c
0075     for i = 1:r
0076         if VerticalEdges(i,j) < 1
0077             Vertical(counter,:)=0;
0078             counter = counter + 1;
0079         else
0080             counter = counter + 1;
0081         end
0082         i = i + 1;
0083     end
0084     j = j + 1;
```

```
0085 end
0086
0087 VerticalCheck = zeros(1,2*vertices);
0088 DeletedRow = zeros(1,2*vertices);
0089 counter = 0;
0090 i = 1;
0091 while i <= size(Vertical,1)             % deletes the rows associated with
0092     VerticalCheck(1,:) = Vertical(i,:); % the deleted vertical edges.
0093     if VerticalCheck == DeletedRow;
0094        Vertical(i,:) = [];
0095        i = i - 1;
0096     end
0097     i = i + 1;
0098 end
0099
0100
0101
0102 NumberHorizontalEdges = c*(r+1);
0103 Horizontal = zeros(NumberHorizontalEdges,2*vertices);
0104
0105 i = 1;
0106 j = 1;
0107 for i = 1:(r+1)*c
0108     Horizontal(i,2*i-1) = -1;          % sets up the rigidity matrix for
0109     Horizontal(i,2*i) = 0;             % horizontal edges of the unit grid
0110     Horizontal(i,2*r + 2*i + 1) = 1;
0111     Horizontal(i,2*r + 2*i + 2) = 0;
0112     i = i+1;
0113 end
0114
0115 HorizontalEdges = ones(r+1,c);
0116 if xcoordinate>0
0117     HorizontalEdges(r-ycoordinate+1,xcoordinate) = 0;
0118     HorizontalEdges(r-ycoordinate+1,xcoordinate + 1) = 0;
0119 end
0120
0121 counter = 1;
0122 i = 1;
0123 j = 1;
0124 for j = 1:c                          % makes the rows associated with the
0125     for i = 1:r+1                    % deleted edges zero
0126         if HorizontalEdges(i,j) < 1
0127             Horizontal(counter,:)=0;
```

```
0128             counter = counter + 1;
0129         else
0130             counter = counter + 1;
0131         end
0132         i = i + 1;
0133     end
0134     j = j + 1;
0135 end
0136
0137
0138 HorizontalCheck = zeros(1,2*vertices); % deletes the rows of zeros
0139 DeletedRow = zeros(1,2*vertices);
0140 counter = 0;
0141 i = 1;
0142 while i <= size(Horizontal,1)
0143     HorizontalCheck(1,:) = Horizontal(i,:);
0144     if HorizontalCheck == DeletedRow;
0145        Horizontal(i,:) = [];
0146        i = i - 1;
0147     end
0148     i = i + 1;
0149 end
0150
0151
0152
0153 BraceEdges = sum(sum(M));   %number of braces in Unit Grid
0154 BraceSubMatrixA = zeros(r,2*r);
0155 BraceSubMatrixB = zeros(r,2*r);
0156 i = 1;
0157 for i = 1:r
0158     BraceSubMatrixA(i,2*i-1) = -1;     % sets up submatrix that makes up
0159     BraceSubMatrixA(i,2*i) = 1;        % the rigidity matrix for the brace
0160     BraceSubMatrixB(i,2*i-1) = 1;      % edges if every possible brace
0161     BraceSubMatrixB(i,2*i) = -1;       % was placed
0162     %Brace(i,2*r + 2*i + 3) = 1;
0163     %Brace(i,2*r + 2*i + 4) = -1;
0164     i = i+1;
0165 end
0166
0167 Brace = zeros(r*c,2*vertices);
0168 i = 1;
0169
0170 for i = 1:c
```

```
0171     Brace(r*(i-1) + 1:r*i,(2*r + 2)*(i-1) + 1:(2*r + 2)*(i-1) + 2*r)...
0172         = BraceSubMatrixA;
0173     Brace(r*(i-1) + 1:r*i,(2*r + 5) + (i-1)*(2*r+2):4*r + 4 +...
0174         (i-1)*(2*r+2)) = BraceSubMatrixB;
0175     i = i + 1;
0176 end
0177
0178 counter = 1;
0179 i = 1;
0180 j = 1;
0181 for j = 1:c                          % makes the rows of the brace edges
0182     for i = 1:r                      % that do not exist zero.
0183         if M(i,j) < 1
0184             Brace(counter,:) = 0;
0185             counter = counter + 1;
0186         else
0187             counter = counter + 1;
0188         end
0189         i = i + 1;
0190     end
0191     j = j + 1;
0192 end
0193
0194 BraceCheck = zeros(1,2*vertices);
0195 DeletedRow = zeros(1,2*vertices);
0196 counter = 0;
0197 i = 1;
0198 while i <= size(Brace,1)
0199     BraceCheck(1,:) = Brace(i,:);   % deletes the rows of zeros
0200     if BraceCheck == DeletedRow;
0201        Brace(i,:) = [];
0202        i = i - 1;
0203     end
0204     i = i + 1;
0205 end
0206
0207 RigidityMatrix = [Vertical; Horizontal; Brace];  % add zero to make a
0208                                                  % square matrix
0209 ExtraZeros = zeros((2*vertices) - size(RigidityMatrix,1)-4,2*vertices);
0210 VertexPinning = zeros(3,2*vertices);
0211 VertexPinning(1,1) = 1;              % pin vertices to get rid of rigid
0212 VertexPinning(2,2) = 1;              % motions from the solution space
0213 VertexPinning(3,3) = 1;
```

```
0214 %VertexPinning(4,4) = 1;
0215 RigidityMatrix = [RigidityMatrix; ExtraZeros;VertexPinning];
0216 i = 1;
0217 for i = 1:2
0218     RigidityMatrix(:,2*deletedvertex - 1) = [];
0219     i = i + 1;
0220 end
0221
0222 NullSpace = null(RigidityMatrix,'r');
0223 NullSpace = NullSpace';             % solution
```

Bibliography

[1] Jack Graver, Brigitte Servatius, and Herman Servatius. *Combinatorial rigidity*, volume 2 of *Graduate Studies in Mathematics*. American Mathematical Society, Providence, RI, 1993.

[2] Jack E. Graver. *Counting on frameworks*, volume 25 of *The Dolciani Mathematical Expositions*. Mathematical Association of America, Washington, DC, 2001. Mathematics to aid the design of rigid structures.

[3] L. Lovász. *Combinatorial problems and exercises*. North-Holland Publishing Co., Amsterdam, 1979.

[4] Kenneth H. Rosen, John G. Michaels, Jonathan L. Gross, Jerrold W. Grossman, and Douglas R. Shier, editors. *Handbook of discrete and combinatorial mathematics*. CRC Press, Boca Raton, FL, 2000.

Glossary

Adjacent Sharing an edge, 3

Bipartite Graph A graph that has a partition of its vertex set into two sets so that every edge in the edge set has one endpoint in each set, 1

Brace An edge of length $\sqrt{2}$ that joins opposite vertices of a unit square framework thus making the unit square into a rigid unit square., 15

Brace Graph A bipartite graph that encodes the bracing of a unit grid., 16

Cell , 16

Column , 16

Connected Graph A graph that is not disconnected., 2

Deformation A motion where the distance between at least one pair of vertices in the vertex set is changed., 5

Degrees of Freedom Dimension of the space of infinitesimal motions., 11

Disconnected Graph if the vertex set can be partitioned into two nonempty sets A and B so that no edge has one endpoint in A and the other endpoint in B., 2

Edge A line segment joining two vertices and a rigid framework of unit length., 1

Edges of the Path The edges joining successive vertices in the sequence, 3

embedding A function $\mathbf{p}$ from a vertex set into m-space; $\mathbf{p}$:V $\rightarrow R^m$., 2

endpoints The vertices that are joined by an edge., 1

Framework A graph $\{V, E\}$ combined with an embedding, 2

General Position No two points are equal and no three lie on a line., 9

Graph A graph G consists of a finite set V(G) of points (vertices) and a finite set E(G) of edges and an assignment of an unordered pairs of elements of V(G) to each edge e E(G) called the endpoints of e., 1

Implied Edge An edge where its associated linear equation is a linear combination of equations associated with other edges., 11

Infinitesimal Deformation , 6

Infinitesimal Motion , 6

Infinitesimal Rigid Motion , 6

Infinitesimal Rigidity , 6

Internal Degrees of Freedom The difference in dimensions between the space of infinitesimal motions and infinitesimal rigid motions., 11

length of the path Number of edges of the path., 3

Motion An indexed family of functions., 4

path A sequence between vertices., 2

Rigid Framework Framework whose motions are only rigid motions and admits no deformations., 5

Rigid Motion A motion that preserves all the distances between any pair of vertices in the vertex set., 5

Rigidity Matrix , 12

Row , 16

Unit Grid , 14

Velocity Vector , 5

Vertex A point., 1

Window , 25