THE SPECTRUM OF THE LAPLACIAN OF COUNTABLE GRAPHS

RACHEL ANNE GRAY, DAVID KRBACHIAN, JUAN SALINAS

Abstract. In this work we study the spectrum of the Laplacian operator on countable locally finite regular graphs through both numerical and analytic methods. The regularity of the graph allows us to reduce the problem to study the adjacency operator. Numerically we approach the problem by studying sequences of subgraphs that approach the countable graph. Analytically, we decompose the adjacency matrices to sums of shift operators and their adjoints on $\ell^2(V)$.

1. Introduction

A graph G is defined to be an ordered pair between a vertex set and an edge set (V, E), where V is the set of *nodes* or *vertices* and E is a collection of two element subsets of V, called the *edges* which connect those nodes. When V is a finite set, given an enumeration of the vertices v_i, $i = 1, \dots, n$ we can associate three matrices to each graph G. The *adjacency matrix of* G is defined to be the matrix $A(G) = (a_{ij})$ such that

$$a_{ij} = \begin{cases} 1 & \text{if } \{v_i, v_j\} \in E \\ 0 & \text{if } \{v_i, v_j\} \notin E \end{cases}$$

Note that since the edges are not ordered (or *directed*), $A(G)$ is a symmetric matrix. Similarly, we define *the degree matrix of* G, $D(G) = (d_{ij})$ such that

$$d_{ij} = \begin{cases} \deg(v_i) & \text{if } i = j, \\ 0 & \text{if } i \neq j. \end{cases}$$

$D(G)$ is a diagonal matrix. Using these we define the *Laplacian matrix* of G to be $L(G) = D(G) - A(G)$. From these definitions we see that $L(G)$ is also a symmetric matrix. Furthermore, it is important to note that these matrices depend on the enumeration of G, but changing enumerations only corresponds to the reordering of a basis, implying that the operator defined on $\mathbb{R}^{|V|}$ is well defined.

The Laplacian matrix can be used to study random walks and diffusion on a graphs (see [2] or [1]) as well as other Markov or stochastic processes. Additionally, as graphs are a natural discretization of subsets of Euclidean space, the Laplacian matrix arises as discretization of the Laplacian operator. Thus the Laplacian matrix comes up in the numberical study of partial differential equations, as in [7] or image processing, [8]. As such, the eigenvalues of $L(G)$ become important in understanding these systems. For more examples of such applications please see [5].

As mentioned above the eigenvalues of $L(G)$ are graph invariants that are used in algebraic graph theory to study many different properties of graphs (see [3]). In particular the second smallest eigenvalue encodes a variety of properties of a graph as discussed in [6].

Partially Supported by NSF-DMS 1247679 grant PUMP: Preparing Undergraduates through Mentoring towards PhDs.

In this paper we study the setting when the vertex set V is countably infinite and that G *locally finite* (the degree of every vertex is finite). In this setting, $A(G)$, $D(G)$ and $L(G)$ are operators defined on $\ell^2(V)$, the Hilbert space of square summable functions on the vertex set. Since G is locally finite we know that these operators are bounded as discussed in [4].

Furthermore we assume that the graph is *regular*: the degrees of all the vertices are the same value, r. In this setting, $D(G) = rI$ so the spectrum of $L(G)$ and $A(G)$ differ by a shift. Thus, in this write up, we focus on the spectrum of the adjacency operator since its entries are only 1 or 0.

In Section 2, we discuss numerical results we computed in MATLAB using converging sequence of subgraphs to approximate the spectrum of the graphs. In Section 3 we discuss the analytic derivation of some of these spectra by decomposing the adjacency operators into shift operators. In Appendix A, we recall some results about the spectrum of shift operators on $\ell^2(\mathbb{N})$. In Appendix B we compile our numerical results into one table for easy reference. Finally, in Appendix C we provide the MATLAB code used in the numerical results.

2. Numerical Results

In this section we describe what data we acquired and how we thought of these finite graphs converging into their infinite counterpart. For a table referencing the data, please see Table 1. We first focused by using the d-path graph P_d to discretized $\mathbb{R}$. Specifically, we discretized $\mathbb{R}$ as the infinite line graph

$$\mathbb{Z} := \lim_{d\to\infty} P_d$$

by computing the adjacency matrices for several subgraphs of $\mathbb{R}$. In this particular case, we found that the adjacency matrices were easy to program into MATLAB in order to analyze. We found that the matrices $A(P_d)$ had the set of eigenvalues that converged to the interval $[-2, 2]$ as d increased and noted the interval was symmetric about zero.

A natural extension of this was to discretize $\mathbb{R}^n$ by the n-dimensional grid

$$\mathbb{Z}^n := \lim_{d\to\infty} (P_d \square P_d \square \cdots \square P_d),$$

where $\square$ is the graph product (see Figure 1 or [2]). Where it seemed natural with our d-path graphs, we encountered the question with the grid in how we should number the vertices. We found that numbering the vertices did not affect the eigenvalue distribution of our adjacency matrices. However, it did affect the difficulty of how we would program them in to MATLAB.

Analyzing our adjacency matrices via MATLAB we found that matrices $A(P_d \square P_d)$ had the set of eigenvalues that converged to the interval $[-4, 4]$ as d increased. Notice the interval is, once again, symmetric about zero.

We next studied the infinite ladder graph $L_{m,n}$, or the m by n grid (see Figure 2). Notice that $P_n \square P_n = L_{n,n}$. Here we found that the eigenvalues of the graph converge to the interval $[-3, 3]$. As we increase the number of connected ladders, constructing the lattice grid from previous, the eigenvalues would then start converging to the previous values of -4 and 4.

Next we studied the triangular lattice and found that it was optimal to number our vertices by taking the subgraphs of our grid lattice $P_d \square P_d$ along with adjoining diagonal vertices, as in Figure 3. After programming the adjacency matrices in MATLAB we found out that the

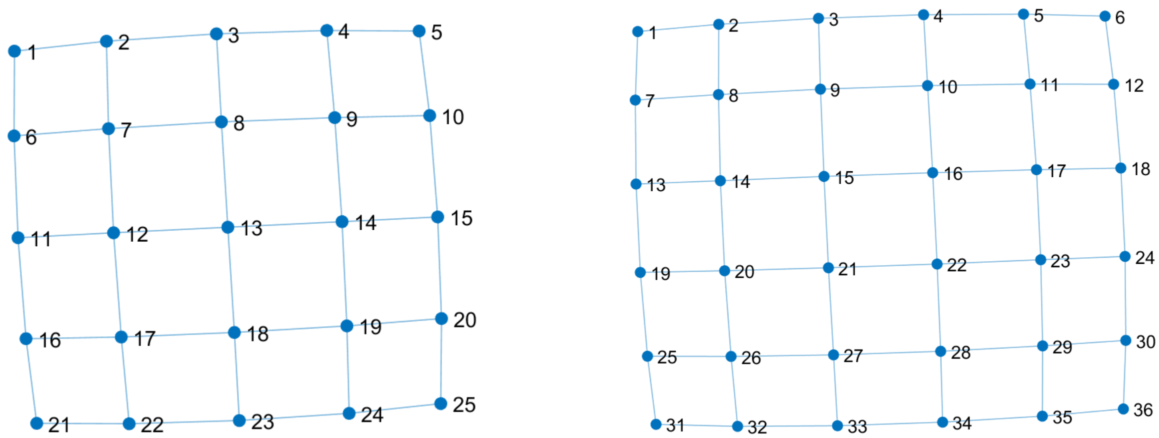


FIGURE 1. The two graphs above are the graphs $P_5 \square P_5$ (on the left) and $P_6 \square P_6$ (on the right). This displays the way we approximate the infinite graph $\mathbb{Z}^2$.

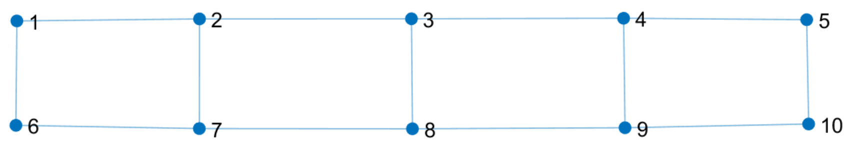


FIGURE 2. Here is our ladder graph $L_{2,5}$ plotted in MATLAB.

eigenvalues converge to the interval $[-3, 6]$. And note, for the first time our interval is not symmetric about zero.

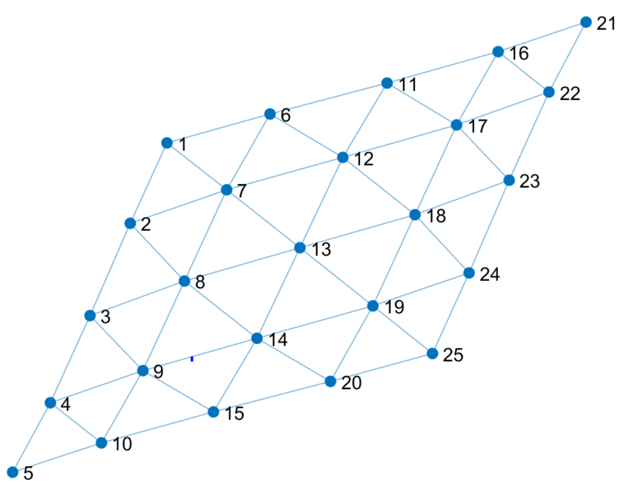


FIGURE 3. This is an approximation of our triangle graph plotted in MATLAB. We extend the graph similarly as in the case with $\mathbb{Z}^2$.

We extended this idea by adjoining both neighboring diagonals of the graph $P_d \square P_d$ (see Figure 4). As we extend this graph, the convergence is the interval $[-4, 8]$.

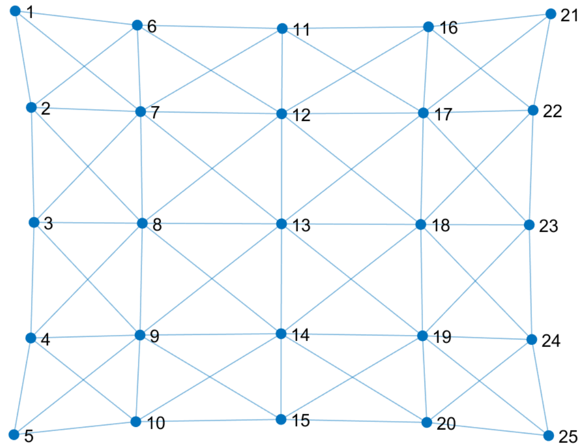


FIGURE 4. This is an approximation of our x-graph plotted in MATLAB. Again, we extend the graph similarly as in the case with $\mathbb{Z}^2$ and the triangle graph.

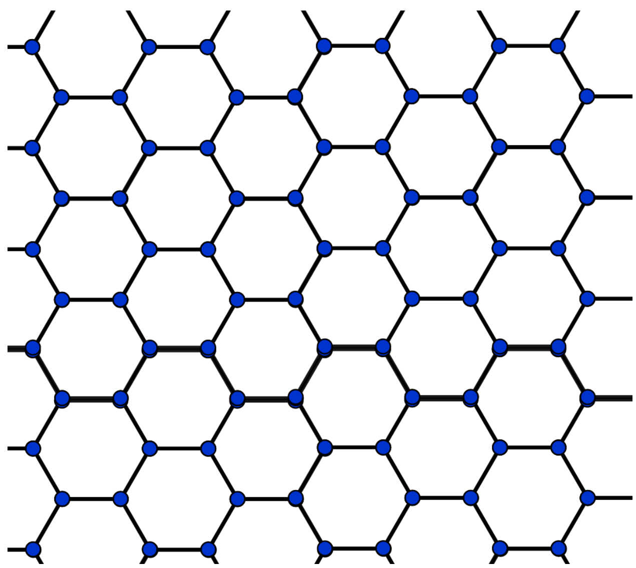


FIGURE 5. The infinite hexagonal graph. We used MATLAB to see that the set of eigenvalues of it's adjacency matrix converged to $[-3, 3]$.

We then focused on the hexagonal lattice (see Figure 5), using the same approach of studying the subgraphs and finding a pattern in their adjacency matrices in order to program them into MATLAB.

Our MATLAB code can be generalized to produce the adjacency matrix $A(P_d \square P_d \square \cdots \square P_d)$. As we let d increase, we found that the eigenvalues of said matrix tended towards the interval $[-2n, 2n]$. As our numerics grew, we began to focus on the connection between our approximations and the eigenvalues of $A(\mathbb{Z}^n)$, and on how to prove our findings rigorously. For this we delved into spectral analysis.

3. Analytical Results

We use topics in algebraic graph theory to study the Laplace differential operator Δ. We study this by discretizing the Laplace equation. That is, we can discretize a space into a graph G and study the Laplacian matrix $L(G)$ defined by $L(G) = D(G) - A(G)$, where $D(G)$ is the degree matrix of G and $A(G)$ is the adjacency matrix of G. In order to make computations easier and more efficient, we restrict G to be an r-regular graph. This then gives the degree matrix of G as $D(G) = rI$, where I is the identity matrix. Hence $L(G) = rI - A(G)$, implying that if λ is an eigenvalue of A, then $r - \lambda$ must be an eigenvalue of L. It is this that allows us to focus on the adjacency matrix of G, then apply this result to find the eigenvalues of L. It is then of interest to think of clever ways to discretize spaces, such as $\mathbb{R}^n$, into graphs.

We define $\ell^2(\mathbb{N})$ to be the space of sequences of $\mathbb{C}$ such that for each $x = (x_1, x_2, \cdots) \in \ell^2(\mathbb{N})$, $\sum |x_i|^2 < \infty$, endowed with the norm $\|x\| := (\sum |x_i|^2)^{1/2}$. We define the left and right shift operators on $\ell^2(\mathbb{N})$ as $S_L, S_R : \ell^2(\mathbb{N}) \to \ell^2(\mathbb{N})$ such that $S_L(x_1, x_2, \cdots) = (x_2, x_3, \cdots)$ and $S_R(x_1, x_2, \cdots) = (0, x_1, \cdots)$, respectively.

Going back to our $\mathbb{Z}$ graph, we see that the infinite tridiagonal matrix $A(\mathbb{Z})$ can be written as:

$$A(\mathbb{Z}) = \begin{pmatrix} 0 & 1 & 0 & 0 & \\ 1 & 0 & 1 & 0 & \cdots \\ 0 & 1 & 0 & 1 & \\ 0 & 0 & 1 & 0 & \ddots \\ & \vdots & & \ddots & \ddots \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 & \\ 1 & 0 & 0 & 0 & \cdots \\ 0 & 1 & 0 & 0 & \\ 0 & 0 & 1 & 0 & \ddots \\ & \vdots & & \ddots & \ddots \end{pmatrix} + \begin{pmatrix} 0 & 1 & 0 & 0 & \\ 0 & 0 & 1 & 0 & \cdots \\ 0 & 0 & 0 & 1 & \\ 0 & 0 & 0 & 0 & \ddots \\ & \vdots & & \ddots & \ddots \end{pmatrix}.$$

Notice that the last two matrices are the matrix representation of our shift operators S_R and S_L, respectively. This tell us that in $\ell^2(\mathbb{N})$, our adjacency operator can be written as $A(\mathbb{Z}) = S_L + S_R$. It is known that S_L and S_R are adjoints of each other, so we have that $A(\mathbb{Z}) = S_L + S_L^*$. Given an operator $\mathcal{L}$ of $\ell^2(\mathbb{N})$, we know that $\lambda \in \sigma(\mathcal{L})$ if and only if $2Re(\lambda) \in \sigma(\mathcal{L} + \mathcal{L}^*)$. Since we found previously that the spectrum of S_L was the closed disk $\sigma(S_L) = \overline{D_1}(0)$ (see appendix A), then this along with the previous theorem yields that $\sigma(A(\mathbb{Z})) = [-2, 2]$.

Afterword

We would like to thank our advisor Dr. Ivan Ventura for both guiding us in the right direction, and for motivating us to explore our mathematical ideas. We would also like to thank Dr. Helena Noronha for arranging the PUMP-URG program.

References

[1] Gary Chartrand and Ortrud R Oellermann. *Applied and algorithmic graph theory*, volume 993. McGraw-Hill New York, 1993.

[2] Gary Chartrand and Ping Zhang. *A first course in graph theory*. Courier Corporation, 2013.

[3] Chris Godsil and Gordon F Royle. *Algebraic graph theory*, volume 207. Springer Science & Business Media, 2013.

[4] Bojan Mohar. The spectrum of an infinite graph. *Linear algebra and its applications*, 48:245–256, 1982.

[5] Bojan Mohar. Some applications of laplace eigenvalues of graphs. In *Graph symmetry*, pages 225–275. Springer, 1997.

[6] Bojan Mohar, Y Alavi, G Chartrand, and OR Oellermann. The laplacian spectrum of graphs. *Graph theory, combinatorics, and applications*, 2(871-898):12, 1991.

[7] Daniel A Spielmat and Shang-Hua Teng. Spectral partitioning works: Planar graphs and finite element meshes. In *Foundations of Computer Science, 1996. Proceedings., 37th Annual Symposium on*, pages 96–105. IEEE, 1996.
[8] Kun Zhou, Jin Huang, John Snyder, Xinguo Liu, Hujun Bao, Baining Guo, and Heung-Yeung Shum. Large mesh deformation using the volumetric graph laplacian. In *ACM transactions on graphics (TOG)*, volume 24, pages 496–503. ACM, 2005.

Appendix A. Spectrum of shift operators

In this section we will review some definition about the spectrum of a bounded linear operator T over a complex Banach space X. Furthermore we remind the reader of a important result of the spectrum of the left shift operator $S_L : \ell^2(\mathbb{N}) \to \ell^2(\mathbb{N})$ and its adjoint the right shift operator S_R.

Definition. Let T be a bounded linear operator over a complex Banach space. The spectrum of T, denoted $\sigma(T)$, is the set of all $\lambda \in \mathbb{C}$ for which the operator $\lambda I - T$ does not have a bounded inverse.

Recall that we define the *left shift operator*, S_L, for $\mathbf{x} = (x_1, x_2, \ldots) \in \ell^2(\mathbb{N})$ by

$$S_L(\mathbf{x}) = S_L(x_1, x_2, \cdots) = (x_2, x_3, \cdots).$$

Similarly, we define the *right shift operator*, S_R, for $\mathbf{x} = (x_1, x_2, \ldots) \in \ell^2(\mathbb{N})$ by

$$S_R(\mathbf{x}) = S_R(x_1, x_2, \cdots) = (0, x_1, \cdots).$$

Lemma 1. *The spectrum of the left and right shift operators are the same:*

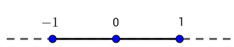
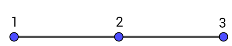
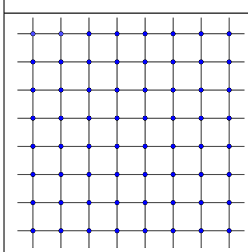
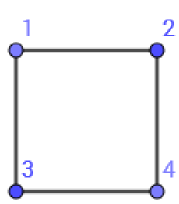
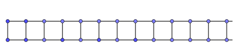
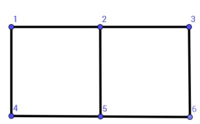
$$\sigma(S_L) = \sigma(S_R) = \{\lambda \in \mathbb{C} \mid |\lambda| \leq 1\}.$$

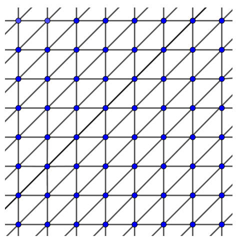
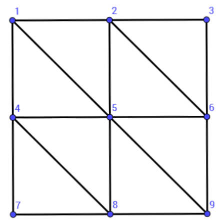
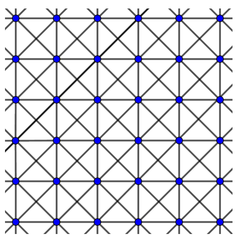
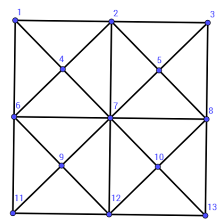
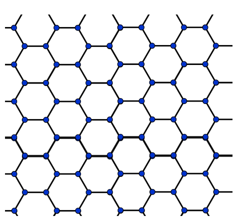
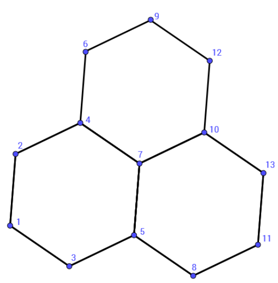
Lemma 2. *The spectrum of $S_L + S_R$ on $\ell^2(\mathbb{N})$ is given by*

$$\sigma(S_L + S_R) = [-2, 2].$$

Appendix B. Table of Data

Table 1

Data Collected			
Infinite Graph	Example Subgraph	Adjacency Matrix of Subgraph	Interval of Convergence
		$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	[-2,2]
		$\begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$	$[-4, 4]$
		$\begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$	$[-3, 3]$

Data cont. 1			
Infinite Graph	Example Subgraph	Adjacency Matrix of Subgraph	Interval of Convergence
		$\begin{bmatrix} 0&1&0&1&1&0&0&0&0\\ 1&0&1&1&1&1&0&0&0\\ 0&1&0&0&1&1&0&0&0\\ 1&1&0&0&1&0&1&1&0\\ 1&1&1&1&0&1&1&1&1\\ 0&1&1&0&1&0&0&1&1\\ 0&0&0&1&1&0&0&1&0\\ 0&0&0&1&1&1&1&0&1\\ 0&0&0&0&1&1&0&1&0 \end{bmatrix}$	$[-3, 6]$
		$\begin{bmatrix} 0&1&0&1&0&1&0&0&0&0&0&0&0\\ 1&0&1&1&1&0&1&0&0&0&0&0&0\\ 0&1&0&0&1&0&0&1&0&0&0&0&0\\ 1&1&0&0&0&1&1&0&0&0&0&0&0\\ 0&1&1&0&0&0&1&1&0&0&0&0&0\\ 1&0&0&1&0&0&1&0&1&0&1&0&0\\ 0&1&0&1&1&1&0&1&1&1&0&1&0\\ 0&0&1&0&1&0&1&0&0&1&0&0&1\\ 0&0&0&0&0&1&1&0&0&0&1&1&0\\ 0&0&0&0&0&0&1&1&0&0&0&1&1\\ 0&0&0&0&0&1&0&0&1&0&0&1&0\\ 0&0&0&0&0&0&1&0&1&1&1&0&1\\ 0&0&0&0&0&0&0&1&0&1&0&1&0 \end{bmatrix}$	$[-4, 8]$
		$\begin{bmatrix} 0&1&1&0&0&0&0&0&0&0&0&0&0\\ 1&0&0&1&0&0&0&0&0&0&0&0&0\\ 1&0&0&0&1&0&0&0&0&0&0&0&0\\ 0&1&0&0&0&1&1&0&0&0&0&0&0\\ 0&0&1&0&0&0&1&1&0&0&0&0&0\\ 0&0&0&1&0&0&0&0&1&0&0&0&0\\ 0&0&0&1&1&0&0&0&0&1&0&0&0\\ 0&0&0&0&1&0&0&0&0&0&1&0&0\\ 0&0&0&0&0&1&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&1&0&0&0&0&1&1\\ 0&0&0&0&0&0&1&1&0&0&0&0&1\\ 0&0&0&0&0&0&0&1&1&1&1&0&1\\ 0&0&0&0&0&0&0&0&0&1&1&0&0 \end{bmatrix}$	$[-6, 6]$

Appendix C. Source-Code

```
%% Function creates adjacency matrix of
%grid of side-length "length" of dimension "dimension".
function Ls=adjacencyLSN(length,dimension)
s=dimension;
n=length;
```

```
nOnes=ones(n^2,1);
A=diag(nOnes(1:n-1), -1) + diag(nOnes(1:n-1), 1);

if s>1
for j=2:s
        nNewOnes=ones(n^j,1);
        I=diag(nNewOnes(1:n^j - n^(j-1)), n^(j-1));
        As=A;
                for k=1:n-1
                        As=blkdiag(As,A);
                end
        A=As+I+I';
        end
end

Ls=A;
end




%Makes adjacency matrix of coordinate axes
function A=coord_line(lines,nodesPerLine)
l=lines;
n=nodesPerLine;
Z=zeros( (n-1)*l+1 );

for i=2:l
        Z(i,1)=1;
end

for i=l+1 : l*(n-1)+1
        Z(i,i-l)=1;
end
A=Z+Z';

end


% Creates adjacency matrix of "lines" amount of lines
%with "nodesPerline" nodes connected as a latter.
function big_A=latter(lines,nodesPerLine)
```

```
n=nodesPerLine;
l=lines;
A=adjacencyL1(n);
As=A;

for i=1:l
if i<l
        As=blkdiag(As,A);
for j=1:n
        As(j,i*n+j)=1;
        As(i*n+j,j)=1;
end
end


end

big_A=As;
end




%Makes adjancency of grid with triangles
function bigA=adjacencyL2withCross(n)
nOnes=ones(n^2,1);
A=adjacencyL1(n);

I=diag(nOnes(1:n^2-n), n);
As=A;
vOnes=ones(n,1);
crossMat=zeros(n^2);

for i=1:n-1
        As=blkdiag(As,A);
end

j=1;
for i=1:n^2-n-1
if mod(i,n)==0
        vOnes(i,1)=0;
end
crossMat(i+n+1,i)=vOnes(j,1);
if mod(i,n)==0
```

```
        j=mod(i,n);
end
j=j+1;
end

bigA=As+I+I'+crossMat+crossMat';
end



%Makes adjancency of grid with both diagonals
function bigA=adjacencyL2withCross(n)
nOnes=ones(n^2,1);
A=adjacencyL1(n);

I=diag(nOnes(1:n^2-n), n);
As=A;
vOnes=ones(n,1);
crossMat=zeros(n^2);
N=crossMat;
for i=1:n-1
        As=blkdiag(As,A);
end

j=1;
for i=1:n^2-n-1
if mod(i,n)==0
        vOnes(i,1)=0;
end
crossMat(i+n+1,i)=vOnes(j,1);
N(i+n,i+1)=crossMat(i+n+1,i);
if mod(i,n)==0
        j=mod(i,n);
end
j=j+1;
end

bigA=As+I+I'+crossMat+crossMat'+N'+N;
end
```