The Laplacian Matrices of Graphs

Daniel A. Spielman

Yale Institute for Network Science

ISIT, July 12, 2016
Outline

Laplacians
  Interpolation on graphs
  Resistor networks
  Spring networks
  Graph drawing
  Clustering
  Linear programming

Sparsification

Solving Laplacian Equations
  Best results
  The simplest algorithm
Interpolate values of a function at all vertices from given values at a few vertices.

Minimize \[ \sum_{(a,b) \in E} (x(a) - x(b))^2 \]

Subject to given values
Interpolation on Graphs

Interpolate values of a function at all vertices from given values at a few vertices.

Minimize \[ \sum_{(a,b) \in E} (x(a) - x(b))^2 \]

Subject to given values
Interpolate values of a function at all vertices from given values at a few vertices.

Minimize \[ \sum_{(a,b) \in E} (x(a) - x(b))^2 = x^T L x \]

Subject to given values

Interpolation on Graphs (Zhu, Ghahramani, Lafferty ’03)
Interpolation on Graphs (Zhu, Ghahramani, Lafferty ’03)

Interpolate values of a function at all vertices from given values at a few vertices.

Minimize \[
\sum_{(a,b) \in E} (x(a) - x(b))^2 = x^T L x
\]

Subject to given values

\[
X(a, b) = 0.51, 0.30, 0.53, 0.61
\]

Take derivatives. Minimize by solving Laplacian
The Laplacian Quadratic Form of $G = (V, E)$

$$x : V \rightarrow \mathbb{R} \quad \sum_{(a,b) \in E} (x(a) - x(b))^2$$
The Laplacian Quadratic Form of $G = (V, E)$

$$x : V \rightarrow \mathbb{R} \quad \sum_{(a,b) \in E} (x(a) - x(b))^2$$
Graphs with positive edge weights

\[
\sum_{(a,b) \in E} w_{a,b} (x(a) - x(b))^2 = x^T L_G x
\]
Resistor Networks

View edges as resistors connecting vertices.

Apply voltages at some vertices. Measure induced voltages and current flow.
Resistor Networks

Induced voltages minimize subject to constraints. \[
\sum_{(a,b) \in E} (x(a) - x(b))^2
\]
Resistor Networks

Induced voltages minimize subject to constraints.

\[ \sum_{(a,b) \in E} (x(a) - x(b))^2 \]
Resistor Networks

Induced voltages minimize subject to constraints.

\[ \sum_{(a,b) \in E} (x(a) - x(b))^2 \]
Resistor Networks

Induced voltages minimize subject to constraints.

\[ \sum_{(a,b) \in E} (x(a) - x(b))^2 \]

Effective resistance = \(1/(\text{current flow at one volt})\)
Spring Networks

View edges as rubber bands or ideal linear springs

Nail down some vertices, let rest settle

When stretched to length $\ell$

potential energy is $\frac{\ell^2}{2}$
Nail down some vertices, let rest settle

Physics: position minimizes total potential energy

\[ \frac{1}{2} \sum_{(a,b) \in E} (x(a) - x(b))^2 \]

subject to boundary constraints (nails)
Drawing by Spring Networks (Tutte ’63)
Drawing by Spring Networks

(Tutte '63)
Drawing by Spring Networks  
(Tutte ’63)
Drawing by Spring Networks

(Tutte ’63)
Drawing by Spring Networks

(Tutte ’63)
If the graph is planar, then the spring drawing has no crossing edges!
Drawing by Spring Networks  
(Tutte ’63)
Drawing by Spring Networks (Tutte '63)
Drawing by Spring Networks (Tutte '63)
Drawing by Spring Networks (Tutte '63)
A $n$-by-$n$ symmetric matrix has $n$ real eigenvalues $\lambda_1 \leq \lambda_2 \cdots \leq \lambda_n$ and eigenvectors $v_1, \ldots, v_n$ such that

$$Lv_i = \lambda_i v_i$$

These eigenvalues and eigenvectors tell us a lot about a graph!
A $n$-by-$n$ symmetric matrix has $n$ real eigenvalues $\lambda_1 \leq \lambda_2 \cdots \leq \lambda_n$ and eigenvectors $v_1, \ldots, v_n$ such that

$$L v_i = \lambda_i v_i$$

These eigenvalues and eigenvectors tell us a lot about a graph!

(excluding $\lambda_1 = 0, v_2 = 1$)
Spectral Graph Drawing

Original Drawing

(Hall '70)
Spectral Graph Drawing

Plot vertex \( a \) at \((v_2(a), v_3(a))\)
draw edges as straight lines

(Original Drawing) → (Spectral Drawing)
Spectral Graph Drawing

Original Drawing

Spectral Drawing

(Hall ’70)
Spectral Graph Drawing

Original Drawing  \rightarrow  Spectral Drawing

(Hall ’70)
Dodecahedron

Best embedded by first three eigenvectors
Erdos’s co-authorship graph
When there is a “nice” drawing

Most edges are short
Vertices are spread out and don’t clump too much

\[ \lambda_2 \text{ is close to 0} \]

When \( \lambda_2 \) is big, say \( \lambda_2 > \frac{10}{|V|^{1/2}} \)

there is no nice picture of the graph
Measuring boundaries of sets

Boundary: edges leaving a set
Measuring boundaries of sets

Boundary: edges leaving a set

Characteristic Vector of \( S \):

\[
x(a) = \begin{cases} 
1 & a \text{ in } S \\
0 & a \text{ not in } S 
\end{cases}
\]
Measuring boundaries of sets

Boundary: edges leaving a set

Characteristic Vector of $S$:

$$x(a) = \begin{cases} 
1 & a \text{ in } S \\
0 & a \text{ not in } S
\end{cases}$$

$$\sum_{(a,b) \in E} (x(a) - x(b))^2 = |\text{boundary}(S)|$$
Spectral Clustering and Partitioning

Find large sets of small boundary

Heuristic to find $x$ with $x^T L_G x$ small

Compute eigenvector $L_G v_2 = \lambda_2 v_2$

Consider the level sets
Spectral Partitioning

(Donath-Hoffman ‘72, Barnes ‘82, Hagen-Kahng ’92)

\[ S = \{ a : v_2(a) \leq t \} \text{ for some } t \]

Cheeger’s inequality implies good approximation
Spectral Partitioning

(Donath-Hoffman '72, Barnes '82, Hagen-Kahng '92)

\[ S = \{ a : v_2(a) \leq t \} \text{ for some } t \]

Cheeger’s inequality implies good approximation
The Laplacian Matrix of a Graph

\[
\begin{pmatrix}
3 & -1 & -1 & -1 & 0 & 0 \\
-1 & 2 & 0 & 0 & 0 & \textcircled{-1} \\
-1 & 0 & 3 & -1 & -1 & 0 \\
-1 & 0 & -1 & 4 & -1 & -1 \\
0 & 0 & -1 & -1 & 3 & -1 \\
0 & \textcircled{-1} & 0 & -1 & -1 & 3
\end{pmatrix}
\]

Symmetric
Non-positive off-diagonals
Diagonally dominant
The Laplacian Matrix of a Graph

\[
x^T L_G x = \sum_{(a,b) \in E} w_{a,b} (x(a) - x(b))^2
\]

\[
L_G = \sum_{(a,b) \in E} w_{a,b} L_{a,b}
\]

\[
L_{1,2} = \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}
\]

\[
= \begin{pmatrix} 1 \\ -1 \end{pmatrix} \begin{pmatrix} 1 & -1 \end{pmatrix}
\]
Quickly Solving Laplacian Equations

S, Teng '04: Using low-stretch trees and sparsifiers

\[ O(m \log^c n \log \epsilon^{-1}) \]

Where \( m \) is number of non-zeros and \( n \) is dimension
Quickly Solving Laplacian Equations

S,Teng ’04: Using low-stretch trees and sparsifiers

\[ O(m \log^c n \log \epsilon^{-1}) \]

Koutis, Miller, Peng ’11: Low-stretch trees and sampling

\[ \tilde{O}(m \log n \log \epsilon^{-1}) \]

Where \( m \) is number of non-zeros and \( n \) is dimension
Quickly Solving Laplacian Equations

S, Teng ’04: Using low-stretch trees and sparsifiers

\[ O(m \log^c n \log \epsilon^{-1}) \]

Koutis, Miller, Peng ’11: Low-stretch trees and sampling

\[ \tilde{O}(m \log n \log \epsilon^{-1}) \]

Cohen, Kyng, Pachocki, Peng, Rao ’14:

\[ \tilde{O}(m \log^{1/2} n \log \epsilon^{-1}) \]

Where \( m \) is number of non-zeros and \( n \) is dimension
Quickly Solving Laplacian Equations

S,Teng ’04: Using low-stretch trees and sparsifiers

\[ O(m \log^c n \log \epsilon^{-1}) \]

Koutis, Miller, Peng ’11: Low-stretch trees and sampling

\[ \tilde{O}(m \log n \log \epsilon^{-1}) \]

Cohen, Kyng, Pachocki, Peng, Rao ’14:

\[ \tilde{O}(m \log^{1/2} n \log \epsilon^{-1}) \]

Good code:

LAMG (lean algebraic multigrid) – Livne-Brandt
CMG (combinatorial multigrid) – Koutis
Quickly Solving Laplacian Equations

S,Teng ’04: Using low-stretch trees and sparsifiers

\[ O(m \log^c n \log \epsilon^{-1}) \]

An \( \epsilon \)-accurate solution to \( L_G x = b \) is an \( \tilde{x} \) satisfying

\[ \| \tilde{x} - x \|_{L_G} \leq \epsilon \| x \|_{L_G} \]

where \( \| v \|_{L_G} = \sqrt{v^T L_G v} = \| L_G^{1/2} v \| \)
Quickly Solving Laplacian Equations

S,Teng ’04: Using low-stretch trees and sparsifiers

\[ O(m \log^c n \log \epsilon^{-1}) \]

An \( \epsilon \)-accurate solution to \( L_G x = b \) is an \( \tilde{x} \) satisfying

\[ \|\tilde{x} - x\|_{L_G} \leq \epsilon \|x\|_{L_G} \]

Allows fast computation of eigenvectors corresponding to small eigenvalues.
Laplacians in Linear Programming

Laplacians appear when solving Linear Programs on graphs by Interior Point Methods

Maximum and Min-Cost Flow  (Daitch, S ’08, Mądry ‘13)

Shortest Paths  (Cohen, Mądry, Sankowski, Vladu ‘16)

Isotonic Regression  (Kyng, Rao, Sachdeva ‘15)

Lipschitz Learning : regularized interpolation on graphs  (Kyng, Rao, Sachdeva, S ‘15)
maximize $f^{out}(s)$

subject to

$f^{out}(a) = f^{in}(a)$, $\forall a \notin \{s, t\}$

$0 \leq f(a, b) \leq c(a, b)$, $\forall (a, b) \in E$
maximize $f^{out}(s)$

subject to

$f^{out}(a) = f^{in}(a), \quad \forall a \notin \{s, t\}$

$0 \leq f(a, b) \leq c(a, b), \quad \forall (a, b) \in E$
Maximize \( f^{out}(s) \)

Subject to \( f^{out}(a) = f^{in}(a), \quad \forall a \notin \{s, t\} \)

\[ 0 \leq f(a, b) \leq c(a, b), \quad \forall (a, b) \in E \]

Multiple calls with varying weights \( w_{a,b} \)

Maximize \( f^{out}(s) \)

Subject to \( f^{out}(a) = f^{in}(a), \quad \forall a \notin \{s, t\} \)

\[ \sum_{(a,b) \in E} w_{a,b} f(a, b)^2 \leq C \]
Spectral Sparsification

Every graph can be approximated by a sparse graph with a similar Laplacian
Approximating Graphs

A graph $H$ is an $\epsilon$-approximation of $G$ if

for all $x$

$$\frac{1}{1 + \epsilon} \leq \frac{x^T L_H x}{x^T L_G x} \leq 1 + \epsilon$$

$$L_H \approx_\epsilon L_G$$
Approximating Graphs

A graph $H$ is an $\epsilon$-approximation of $G$ if

for all $x$

$$\frac{1}{1 + \epsilon} \leq \frac{x^T L_H x}{x^T L_G x} \leq 1 + \epsilon$$

Preserves boundaries of every set

(1 $\pm$ $\epsilon$)
Solutions to linear equations are similar.

As are effective resistances.

Approximating Graphs

A graph \( H \) is an \( \epsilon \)-approximation of \( G \) if

for all \( x \)

\[
\frac{1}{1 + \epsilon} \leq \frac{x^T L_H x}{x^T L_G x} \leq 1 + \epsilon
\]

Solutions to linear equations are similar

\[
L_H \approx_\epsilon L_G \iff L_H^{-1} \approx_\epsilon L_G^{-1}
\]

As are effective resistances.
Yield good LDPC codes

Every set of vertices has large boundary

$\lambda_2$ is large

Random regular graphs are usually expanders
Sparsification by Random Sampling

Assign a probability \( p_{a,b} \) to each edge \((a,b)\)

Include edge \((a,b)\) in \(H\) with probability \( p_{a,b} \).

If include edge \((a,b)\), give it weight \( w_{a,b} / p_{a,b} \)

\[
\mathbb{E} \left[ L_H \right] = \sum_{(a,b) \in E} p_{a,b} \left( w_{a,b} / p_{a,b} \right) L_{a,b} = L_G
\]
Choose $p_{a,b}$ to be $w_{a,b}$ times the effective resistance between $a$ and $b$.

Low resistance between $a$ and $b$ means there are many alternate routes for current to flow and that the edge is not critical.

Proof by random matrix concentration bounds (Rudelson, Ahlswede-Winter, Tropp, etc.)

Only need $O(n \log n/\epsilon^2)$ edges

(S, Srivastava `08)
Optimal Graph Sparsification?

For every $G = (V, E, w)$, there is a $H = (V, F, z)$ s.t.

$$L_G \approx_\epsilon L_H \quad \text{and} \quad |F| \leq (2 + \epsilon)^2 n / \epsilon^2$$

Is within a factor of 2 of how well
Ramanujan expanders approximate complete graphs

(Batson, S, Srivastava '09)
Approximate Gaussian Elimination

(Kyng & Sachdeva ‘16)

Gaussian Elimination: compute upper triangular \( U \) so that

\[
L_G = U^T U
\]

Approximate Gaussian Elimination: compute sparse upper triangular \( U \) so that

\[
L_G \approx U^T U
\]
Additive view of Gaussian Elimination

Find $U$, upper triangular matrix, s.t $U^T U = A$

\[
A = \begin{pmatrix}
16 & -4 & -8 & -4 \\
-4 & 5 & 0 & -1 \\
-8 & 0 & 14 & 0 \\
-4 & -1 & 0 & 7
\end{pmatrix}
\]
Additive view of Gaussian Elimination

\[
\begin{pmatrix}
16 & -4 & -8 & -4 \\
-4 & 5 & 0 & -1 \\
-8 & 0 & 14 & 0 \\
-4 & -1 & 0 & 7
\end{pmatrix}
\]

Find the rank-1 matrix that agrees on the first row and column.

\[
\begin{pmatrix}
16 & -4 & -8 & -4 \\
-4 & 1 & 2 & 1 \\
-8 & 2 & 4 & 2 \\
-4 & 1 & 2 & 1
\end{pmatrix}
= \begin{pmatrix} 4 \\ -1 \end{pmatrix} \begin{pmatrix} 4 \\ -1 \end{pmatrix}^T
\]
Additive view of Gaussian Elimination

\[
\begin{pmatrix}
16 & -4 & -8 & -4 \\
-4 & 5 & 0 & -1 \\
-8 & 0 & 14 & 0 \\
-4 & -1 & 0 & 7 \\
\end{pmatrix}
- \\
\begin{pmatrix}
16 & -4 & -8 & -4 \\
-4 & 1 & 2 & 1 \\
-8 & 2 & 4 & 2 \\
-4 & 1 & 2 & 1 \\
\end{pmatrix}
\]

Subtract the rank 1 matrix.
We have eliminated the first variable.
Additive view of Gaussian Elimination

\[
\begin{pmatrix}
0 & 0 & 0 & 0 \\
0 & 4 & -2 & -2 \\
0 & -2 & 10 & -2 \\
0 & -2 & -2 & 6 \\
\end{pmatrix}
\]
Additive view of Gaussian Elimination

\[
\begin{pmatrix}
0 & 0 & 0 & 0 \\
0 & 4 & -2 & -2 \\
0 & -2 & 10 & -2 \\
0 & -2 & -2 & 6
\end{pmatrix}
\]

Find the rank-1 matrix that agrees on the next row and column.

\[
\begin{pmatrix}
0 & 0 & 0 & 0 \\
0 & 4 & -2 & -2 \\
0 & -2 & 1 & 1 \\
0 & -2 & 1 & 1
\end{pmatrix}
= \begin{pmatrix} 0 \\ 2 \\ 2 \end{pmatrix}^T \begin{pmatrix} 0 \\ -1 \\ -1 \end{pmatrix}
\]
Additive view of Gaussian Elimination

\[
\begin{pmatrix}
0 & 0 & 0 & 0 \\
0 & 4 & -2 & -2 \\
0 & -2 & 10 & -2 \\
0 & -2 & -2 & 6
\end{pmatrix}
\begin{pmatrix}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 9 \\
0 & 0 & 0 & -3
\end{pmatrix}
\]

Subtract the rank 1 matrix.
We have **eliminated** the second variable.
Additive view of Gaussian Elimination

\[
A = \begin{pmatrix}
16 & -4 & -8 & -4 \\
-4 & 5 & 0 & -1 \\
-8 & 0 & 14 & 0 \\
-4 & -1 & 0 & 7
\end{pmatrix}
\]

\[
= \begin{pmatrix}
4 \\
-1 \\
-2 \\
-1
\end{pmatrix}
\begin{pmatrix}
4 \\
-1 \\
-2 \\
-1
\end{pmatrix}^T + \begin{pmatrix}
0 \\
2 \\
-1 \\
-1
\end{pmatrix}
\begin{pmatrix}
0 \\
2 \\
-1 \\
-1
\end{pmatrix}^T + \begin{pmatrix}
0 \\
0 \\
3 \\
-1
\end{pmatrix}
\begin{pmatrix}
0 \\
0 \\
3 \\
-1
\end{pmatrix}^T + \begin{pmatrix}
0 \\
0 \\
2 \\
2
\end{pmatrix}
\begin{pmatrix}
0 \\
0 \\
2 \\
2
\end{pmatrix}^T
\]

Running time proportional to sum of squares of number of non-zeros in these vectors.
Additive view of Gaussian Elimination

\[ A = \begin{pmatrix} 16 & -4 & -8 & -4 \\ -4 & 5 & 0 & -1 \\ -8 & 0 & 14 & 0 \\ -4 & -1 & 0 & 7 \end{pmatrix} = \begin{pmatrix} 4 \\ -1 \\ -2 \\ -1 \end{pmatrix} \begin{pmatrix} 4 \end{pmatrix}^T + \begin{pmatrix} 0 \\ 2 \\ -1 \\ -1 \end{pmatrix} \begin{pmatrix} 2 \end{pmatrix}^T + \begin{pmatrix} 0 \\ 0 \\ 3 \\ -1 \end{pmatrix} \begin{pmatrix} 3 \end{pmatrix}^T + \begin{pmatrix} 0 \\ 0 \\ 2 \end{pmatrix} \begin{pmatrix} 2 \end{pmatrix}^T = \begin{pmatrix} 4 & 0 & 0 & 0 \\ -1 & 2 & 0 & 0 \\ -2 & -1 & 3 & 0 \\ -1 & -1 & -1 & 2 \end{pmatrix} \begin{pmatrix} 4 & -1 & -2 & -1 \\ 0 & 2 & -1 & -1 \\ 0 & 0 & 3 & -1 \\ 0 & 0 & 0 & 2 \end{pmatrix} \]
Additive view of Gaussian Elimination

\[ A = \begin{pmatrix} 16 & -4 & -8 & -4 \\ -4 & 5 & 0 & -1 \\ -8 & 0 & 14 & 0 \\ -4 & -1 & 0 & 7 \end{pmatrix} \]

\[ = \begin{pmatrix} 4 \\ -1 \\ -2 \\ -1 \end{pmatrix} \begin{pmatrix} 4 \end{pmatrix}^T + \begin{pmatrix} 0 \\ 2 \\ -1 \\ -1 \end{pmatrix} \begin{pmatrix} 0 \end{pmatrix}^T + \begin{pmatrix} 0 \\ 0 \\ 3 \\ -1 \end{pmatrix} \begin{pmatrix} 0 \end{pmatrix}^T + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 2 \end{pmatrix} \begin{pmatrix} 0 \end{pmatrix}^T \]

\[ = \begin{pmatrix} 4 & -1 & -2 & -1 \\ 0 & 2 & -1 & -1 \\ 0 & 0 & 3 & -1 \\ 0 & 0 & 0 & 2 \end{pmatrix}^T \begin{pmatrix} 4 & -1 & -2 & -1 \\ 0 & 2 & -1 & -1 \\ 0 & 0 & 3 & -1 \\ 0 & 0 & 0 & 2 \end{pmatrix} = U^T U \]
Gaussian Elimination of Laplacians

If this is a Laplacian,

\[
\begin{pmatrix}
16 & -4 & -8 & -4 \\
-4 & 5 & 0 & -1 \\
-8 & 0 & 14 & 0 \\
-4 & -1 & 0 & 7
\end{pmatrix}
- \begin{pmatrix}
4 \\
-1 \\
-2 \\
-1
\end{pmatrix}
\begin{pmatrix}
4 \\
-1 \\
-2 \\
-1
\end{pmatrix}^T
= \begin{pmatrix}
0 & 0 & 0 & 0 \\
0 & 4 & -2 & -2 \\
0 & -2 & 10 & -2 \\
0 & -2 & -2 & 6
\end{pmatrix}
\]

then so is this
Gaussian Elimination of Laplacians

If this is a Laplacian,

\[
\begin{pmatrix}
16 & -4 & -8 & -4 \\
-4 & 5 & 0 & -1 \\
-8 & 0 & 14 & 0 \\
-4 & -1 & 0 & 7
\end{pmatrix}
- \begin{pmatrix}
4 \\
-1 \\
-2 \\
-1
\end{pmatrix}
\begin{pmatrix}
4 \\
-1 \\
-2 \\
-1
\end{pmatrix}^T
= \begin{pmatrix}
0 & 0 & 0 & 0 \\
0 & 4 & -2 & -2 \\
0 & -2 & 10 & -2 \\
0 & -2 & -2 & 6
\end{pmatrix}
\]

When eliminate a node, add a clique on its neighbors
Approximate Gaussian Elimination

1. when eliminate a node, add a clique on its neighbors

2. Sparsify that clique, without ever constructing it

(Kyng & Sachdeva ‘16)
Approximate Gaussian Elimination

(Kyng & Sachdeva ‘16)

1. When eliminate a node of degree $d$,
   add $d$ edges at random between its neighbors, sampled with probability proportional to the weight of the edge to the eliminated node.
Approximate Gaussian Elimination

(Kyng & Sachdeva ‘16)

0. Initialize by randomly permuting vertices, and making \(O(\log^2 n)\) copies of every edge

1. When eliminate a node of degree \(d\), add \(d\) edges at random between its neighbors, sampled with probability proportional to the weight of the edge to the eliminated node

Total time is \(O(m \log^3 n)\)
Approximate Gaussian Elimination

(Kyng & Sachdeva ‘16)

0. Initialize by randomly permuting vertices, and making $O(\log^2 n)$ copies of every edge

1. When eliminate a node of degree $d$, add $d$ edges at random between its neighbors, sampled with probability proportional to the weight of the edge to the eliminated node

Total time is $O(m \log^3 n)$

Can be improved by sacrificing some simplicity
Approximate Gaussian Elimination

(Kyng & Sachdeva ‘16)

Analysis by Random Matrix Theory:

Write $U^T U$ as a sum of random matrices.

$\mathbb{E} [U^T U] = L_G$

Random permutation and copying control the variances of the random matrices.

Apply Matrix Freedman inequality (Tropp ‘11)
Recent Developments

Other families of linear systems
(Kyng, Lee, Peng, Sachdeva, S '16)

complex-weighted Laplacians
\[
\begin{pmatrix}
1 & e^{i\theta} \\
e^{-i\theta} & 1
\end{pmatrix}
\]

connection Laplacians
\[
\begin{pmatrix}
I & Q \\
Q^T & I
\end{pmatrix}
\]

Laplacians.jl
To learn more

My web page on:

Laplacian linear equations, sparsification, local graph clustering, low-stretch spanning trees, and so on.

My class notes from

“Graphs and Networks” and “Spectral Graph Theory”

$Lx = b$, by Nisheeth Vishnoi