МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«Дальневосточный федеральный университет»
ИНСТИТУТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ
ТЕХНОЛОГИЙ
ДЕПАРТАМЕНТ МАТЕМАТИЧЕСКОГО И КОМПЬЮТЕРНОГО
МОДЕЛИРОВАНИЯ

УДК _____
№ госрегистрации _____
Инв. №_____

УТВЕРЖДАЮ

_____
головной исполнитель НИР

_____ _____

«_____» _____ 2024 г.

ОТЧЁТ
О НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ

Алгоритмическая теория графов

по теме:

"Случайные блуждания: связь с резистивным расстоянием"

(заключительный)

Revision: 1.27, Date: 2024/07/15 12:59:36, Author: lunk

Руководитель темы _____ Е.А. Нурминский

г. Владивосток, 2024

Список исполнителей

Вершинин Д.А. _____ подпись

Пелагеев Д.И. _____ подпись

Содержание

ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ

В настоящем отчете о НИР применяют следующие термины с соответствующими определениями.

Электрическая цепь — Набор электрических компонентов, таких как резисторы, конденсаторы и источники питания, соединённых проводниками для создания замкнутого пути, по которому может течь электрический ток.

Граф — Математическая структура, состоящая из вершин (узлов) V и рёбер E, соединяющих пары вершин, используемая для моделирования различных систем и их взаимосвязей.

Вершина — Фундаментальный элемент в теории графов, представляющий объект или точку соединения в электрической цепи или другой сети. Также иногда называется узлом.

Ребро — Соединение между двумя вершинами графа, вес которого соответствует электрическому сопротивлению соединения в электрической цепи.

Смежные вершины — Две вершины называются смежными, если они соединены ребром. Если вершины x и y смежные, то это можно записать как $x \sim y$. Все смежные вершины для вершины x обозначим, как $\sim x$.

Резистивное расстояние — Сопротивление между двумя вершинами в графе, моделирующем электрическую цепь, когда все сопротивления заменены на единичные.

Законы Кирхгофа — Физические законы, описывающие сохранение электрического заряда в электрической цепи. Первый закон (Закон узлов) утверждает, что сумма токов, входящих в узел, равна сумме токов, выходящих из узла. Второй закон (Закон контуров) гласит, что сумма напряжений в любом замкнутом контуре цепи равна нулю.

Матрица смежности — Квадратная матрица, используемая для представления графа, где элементы указывают наличие или отсутствие рёбер между парами вершин.

Лапласиан — Квадратная матрица, вычисляемая на основе матрицы смежности графа, которая используется для анализа свойств графов.

Обратная матрица Лапласиана — Матрица, полученная при обращении Лапласиана графа, используемая для вычисления резистивного расстояния между вершинами графа.

Обращение матрицы — Операция, при которой для данной квадратной матрицы A находится такая матрица A^{-1}, что произведение $A \cdot A^{-1}$ равно единичной матрице.

Случайное блуждание — Модель, описывающая случайное перемещение по вершинам графа, где вероятность перехода из одной вершины в другую определяется весами рёбер.

Сопротивление — Физическая величина, характеризующая способность проводника препятствовать прохождению электрического тока, измеряемая в омах (Ω). Существует несколько способов расчета сопротивления:

— **Закон Ома:** Сопротивление R можно определить как отношение напряжения U на проводнике к силе тока I, протекающего через него: $R = \frac{U}{I}$.

— **Последовательное соединение резисторов:** Общее сопротивление $R_{\text{сум}}$ для последовательного соединения резисторов определяется как сумма сопротивлений всех резисторов: $R_{\text{сум}} = R_1 + R_2 + \ldots + R_n$.

— **Параллельное соединение резисторов:** Общее сопротивление $R_{\text{п}}$ для параллельного соединения резисторов определяется как обратная величина суммы обратных значений сопротивлений всех резисторов: $\frac{1}{R_{\text{п}}} = \frac{1}{R_1} + \frac{1}{R_2} + \ldots + \frac{1}{R_n}$.

Напряжение — Разность электрических потенциалов между двумя точками электрической цепи, определяющая способность источника энергии выполнять работу по перемещению заряда, измеряемая в вольтах (В).

Сила тока — Количество электрического заряда, проходящего через поперечное сечение проводника за единицу времени, измеряемая в амперах (А).

ВВЕДЕНИЕ

Теория графов часто применяется для исследования электрических цепей. Одной ее из задач является нахождение сопротивления между двумя узлами электрической сети. Эту задачу можно рассматривать в контексте теории графов, т.к. электрическую цепь можно моделировать с помощью связного графа, где вершинами графа являются узлы сети. Два узла сети являются смежными, если между ними существует прямая связь. Это означает, что между этими узлами существует физическое соединение. Такому соединению можно сопоставить ребро графа, где его весом является электрическая проводимость между двумя узлами сети, взятых изолированно от остальной сети.

С введением графовой интерпретации можно задать метрику для нахождения расстояния между двумя произвольными вершинами графа – резистивное расстояние[1]. Резистивное расстояние между двумя вершинами простого связного графа G равно сопротивлению между двумя узлами электрической цепи, построенной путем замены сопротивления всех связей сети на сопротивление в 1 Ом. В таком случае вес каждого ребра будет равен единице. В то же время, для электрической цепи применим первый закон Кирхгофа : "Алгебраическая сумма токов в узле электрической цепи равна нулю".

Связь между Законами Кирхгофа и резистивным расстоянием начали исследовать Douglas J. Klein и M. Randic [2] несколько десятилетий назад.

В данной работе рассматривается модель электрической цепи, представленной в виде графа, где вершины графа соответствуют узлам электрической сети, а вес ребра графа — сопротивлениям между этими узлами, взятыми изолированно от основной сети. Главной целью является нахождение резистивного расстояния между двумя произвольными вершинами графа. Это расстояние эквивалентно сопротивлению между соответствующими узлами в электрической цепи. Модель позволяет использовать

математические методы теории графов для анализа электрических цепей, что упрощает вычисления и интерпретацию результатов.

Прикладной задачей является разработка алгоритма для вычисления резистивного расстояния, что открывает возможности для применения этой модели в различных областях, включая оптимизацию маршрутов и анализ сложных сетей.

Существует также тесная связь резистивного расстояния с методом случайных блужданий. Этот метод основывается на предположении, что время перемещения по каждому ребру равно единице, а выбор следующей вершины происходит независимо с равной вероятностью для всех соседей текущей вершины. Случайное блуждание моделирует поведение тока в электрических цепях.

1 Теоретические аспекты электрических сетей

Стоит отметить, что существует два вида закона Ома: линейный и нелинейный.

Линейный закон Ома утверждает, что ток через проводник прямо пропорционален напряжению, приложенному к нему, при условии, что температура и другие физические условия остаются постоянными. Это выражается формулой:

$$I = \frac{U}{R},$$

где I – ток (в амперах, А), U – напряжение (в вольтах, В), R – сопротивление (в омах, Ом).

Нелинейный закон Ома описывает материалы и устройства, у которых нет прямой зависимости тока от напряжения. В таких случаях ток может изменяться при изменении напряжения или сопротивления нелинейно. Формула нелинейного закона Ома часто значительно сложнее, чем линейный закон, и зависит от специфики материала или компонента. Для описания таких зависимостей обычно применяют эмпирические формулы.

В работе рассматривается только линейный закон Ома.

1.1 Физические определения

Определение 1.1.1 (Ток)**.** Ток i_{xy} из вершины x в y графа G определяется на парах смежных вершин так, что:

$$i_{xy} = -i_{yx} \tag{1.1}$$

Ток i_{xy} определяется уравнением:

$$\sum_{y}^{\sim x} i_{xy} = I \cdot \delta_{xa} - I \cdot \delta_{xb}, \ \forall x \in V(G) \tag{1.2}$$

где сумма берется по всем $y \in V(G)$, смежным с вершиной x; δ_{xy} – символ Кронекера.

Формула (1.2) описывает сумму токов, входящих в вершину x. Величина $I\delta_{xa}$ равна I, если $x = a$ (источник), и 0 – в противном случае. Аналогично, $I\delta_{xb}$ равно I, если $x = b$ (сток), и 0 – в противном случае. Таким образом, I представляет собой величину чистого тока, выходящего из источника a и входящего в сток b. Это уравнение известно как закон Кирхгофа для тока (в случае одного источника и одного стока).

Определение 1.1.2 (Физический ток)**.** Ток называют физическим [2], если существует потенциал v на вершинах графа G такой, что:

$$i_{xy} \cdot r_{xy} = v_x - v_y, \ \forall x,y \in E(G), \tag{1.3}$$

где $r_{xy} \equiv r_{yx}$, v_x - потенциал на узле x, v_y - потенциал на узле y.

Потенциал известен как напряжение (аналог давления в гидравлике), а уравнение (1.3) также известно как закон Ома. Для тока в графе G существование такого потенциала может быть показано через второй закон Кирхгофа для цепей:

$$\sum_{x\sim y}^{C} i_{xy} r_{xy} = 0, \ \forall C \tag{1.4}$$

где сумма берется по всем ребрам контура C в графе G, а ток i из упорядоченной последовательности токов контура C.

1.2 Нормированное пространство

В соответствии с графом G можно определить нормированное пространство с ортонормированным базисом, элементы которого сопоставлены с вершинами графа G. Такие базисные вектора определяется как $\mathbf{e}, \mathbf{e} \in V(G)$. Размерность этого пространства равна количеству вершин в графе G. В этом пространстве представлены несколько матриц (линейных операторов).

Определение 1.2.1 (Матрица инцидентности)**.** Матрицей инцидентности называется матрица, элементы которой задаются следующим образом:

$$A_{xy} = \begin{cases} 1/r_{xy}, & x \sim y \\ 0, & otherwise \end{cases} \quad x,y \in V(G), \tag{1.5}$$

Элементы матрицы инцидентности A можно также получить через произведение вектора $\mathbf{x}^T$, матрицы A и вектора $\mathbf{y}$

$$A_{xy} = \mathbf{x}^T A \mathbf{y},$$

где $\mathbf{x}$ - вектор, в котором элемент с номером x равен 1, а остальные 0; $\mathbf{y}$ - вектор,в котором элемент с номером y равен 1, а остальные 0; A - матрица инцидентности графа G.

При установлении единичных сопротивлений в цепи, матрица A сводится к матрице смежности графа G.

Определение 1.2.2 (Матрица степеней)**.** Матрица Δ, элементы которой задаются следующим образом:

$$\Delta_{xy} = \delta_{xy} \sum_{z}^{\sim x} 1/r_{xz},\ x,z \in V(G) \tag{1.6}$$

где суммирование происходит по вершинам z, которые смежны с x, называется матрицей степеней вершин.

Элементы матрицы степеней Δ можно также получить через скалярное произведение вектора $\mathbf{x}^T$, матрицы Δ и вектора $\mathbf{y}$

$$\Delta_{xy} = \mathbf{x}^T \Delta \mathbf{y},$$

где $\mathbf{x}$ - вектор,в котором элемент с номером x равен 1, а остальные 0; $\mathbf{y}$ - вектор,в котором элемент с номером y равен 1, а остальные 0; Δ - матрица степеней графа G.

Далее большую роль сыграет матрица под названием Лапласиан, которая задаётся через разность $\Delta - A$.

Лемма 1.2.3 (О собственных значениях)**.** *Лапласиан имеет неотрицательные собственные значения, минимум одно из которых ноль. Если граф G связный, то соответствующий собственный вектор (с точностью до скалярного множителя) равен:*

$$\phi \equiv \sum_{x} \mathbf{x},\ \forall x \in V(G) \tag{1.7}$$

Для уменьшения размера работы доказательство опускается. Оно представлено в [2] см. 86 стр.

Следствие 1.2.4. *По условиям леммы, Лапласиан имеет как минимум одно собственное значение равное нулю. Следовательно, он не является обратимым в привычном смысле. Однако в подпространстве, ортогональном к ϕ, он имеет псевдообратную матрицу, которая задаётся следующим образом:*

$$(\Delta - A)^+ = ((\Delta - A)^T(\Delta - A))^{-1}(\Delta - A)^T$$

1.3 Резистивное расстояние

Лемма 1.3.1 (О токе)**.** *Физический ток в связном графе G из вершины a в вершину b существует, является единственным и определяется следующим образом:*

$$i_{xy} = \frac{I}{r_{xy}}((\mathbf{x} - \mathbf{y})^T(\Delta - A)^+(\mathbf{a} - \mathbf{b})), \tag{1.8}$$

где $\mathbf{a}, \mathbf{b}, \mathbf{x}$,$\mathbf{y}$ - векторы, в которых элементы с номерами a,b,x,y (соответственно) равны 1, а остальные элементы – 0; $(\Delta - A)^+$ - псевдообратная матрица Лапласиана.

Доказательство. Для доказательства, подставим (1.3) в (1.2) и получим:

$$\sum_{y}^{\sim x} \frac{1}{r_{xy}}\{v_x - v_y\} = I\delta_{xa} - I\delta_{xb} \tag{1.9}$$

Используя (1.6) и (1.5), получим:

$$\mathbf{x}^T\Delta\mathbf{x}v_x - \sum_{y}\mathbf{x}^T A\mathbf{y}v_y = I(\mathbf{x}\cdot(\mathbf{a} - \mathbf{b})), \tag{1.10}$$

где $\mathbf{a}, \mathbf{b}, \mathbf{x}$,$\mathbf{y}$ - векторы, в которых элементы с номерами a,b,x,y (соответственно) равны 1, а остальные 0.

Теперь $\forall x \in V(G)$ и т.к. x и y сопостовимы, перепишем (1.10) как:

$$\Delta\sum_{x}\mathbf{x}v_x - A\sum_{x}\mathbf{x}v_x = I(\mathbf{a} - \mathbf{b})$$

$$(\Delta - A)\sum_x v_x \mathbf{x} = I(\mathbf{a} - \mathbf{b}). \tag{1.11}$$

Тогда, согласно лемме о собственных значениях, (1.11) можно переписать как:

$$\sum_x v_x \mathbf{x} = I(\Delta - A)^+(\mathbf{a} - \mathbf{b}), \tag{1.12}$$

Это непосредственно приводит к формуле нашей теоремы, тем самым устанавливая уникальность этих различий. Для нахождения разности потенциалов между парой вершин x, y нужно из элемента вектора $\sum_x v_x \mathbf{x}$ с номером x вычесть элемент с номером y, что дает нам:

$$v_x - v_y = I((\mathbf{x} - \mathbf{y})^T(\Delta - A)^+(\mathbf{a} - \mathbf{b})) \tag{1.13}$$

Тогда, закон Ома (1.3) дает формулу леммы, единственность I и его существование:

$$v_x - v_y = i_{xy} r_{xy} \Rightarrow i_{xy} = \frac{v_x - v_y}{r_{xy}} = \frac{I}{r_{xy}}((\mathbf{x} - \mathbf{y})^T(\Delta - A)^+(\mathbf{a} - \mathbf{b})) \tag{1.14}$$

□

Разность потенциалов между двумя точками, как видно из закона Ома, прямо пропорциональна i. При выборе $x = a$ и $y = b$, этот коэффициент пропорциональности называется резистивным расстоянием Ω_{ab} между a и b. Таким образом, мы получаем основной результат этого раздела:

Теорема 1.3.2 (О резистивном расстоянии)**.** *Для физического тока из a в b в графе G резистивное расстояние Ω_{ab} равно:*

$$\Omega_{ab} = (\mathbf{a} - \mathbf{b})^T(\Delta - A)^+(\mathbf{a} - \mathbf{b}) \tag{1.15}$$

Теорема 1.3.3 (Сопротивление эквивалентно расстоянию)**.** *Формально определим сопротивления как расстояние. Под расстоянием на графе G мы понимаем отображение ρ из декартова произведения $V(G) \times V(G)$ в вещественные числа, удовлетворяющее следующим аксиомам:*

$$\rho(a,b) \geq 0,$$

$$\rho(a,b) = 0 \Leftrightarrow a = b,$$

$$\rho(a,b) = \rho(b,a),$$

$$\rho(a,x) = \rho(x,b) \geq \rho(a,b)$$

Доказательство приведено в [2] см. 89 стр.

2 Теоретические основы случайных блужданий

Случайное блуждание на графе G представляет собой процесс, при котором на каждом шаге изменяется состояние случайного процесса, заключающееся в перемещении из текущей вершины в одну из соседних вершин, выбранную независимым образом с равной вероятностью.

Цель изучения случайных блужданий заключается в их универсальности и применимости к широкому спектру задач в науке и технике. Они предоставляют мощные инструменты для моделирования, анализа и решения сложных проблем, связанных с неопределенностью и случайностью.

Рассмотрим однородное случайное блуждание, при котором вероятность перехода из вершины x в любую соседнюю вершину одинакова и равна $1/d(x)$, где $d(x)$ - степень вершины x.

Для начала определим сценарий, который будет полезен в доказательстве нескольких теорем.

Определение 2.0.1 (Сценарий А)**.** Подадим на каждый узел сети ток, равный количеству связей этого узла ($d(x)$). Выберем узел y и получим из него ток, равный $2m = \sum_x d(x)$, где m - количество связей в сети. Выбор количества тока обусловлен тем фактом, что сумма количества связей каждой вершины даст число, равное удвоенному количеству связей в цепи.

Определение 2.0.2 (Время попадания)**.** Для любых двух вершин x и y графа G время попадания (hitting time, H_{xy}) определяется, как мат ожидание или же среднее количество шагов (steps) случайного блуждания, необходимое для достижения вершины y из вершины x.

$$H_{xy} = \mathbb{E}[steps]$$

Определение 2.0.3 (Время возвращения)**.** Время возвращения (commute time) C_{xy} между двумя вершинами x и y в графе G равно сумме времени попадания из вершины x в вершину y и времени попадания из y в x.

$$C_{xy} = H_{xy} + H_{yx}$$

Теорема 2.0.4 (Время первого попадания)**.** *При сценарии A разность потенциалов ϕ_{xy} между узлами x и y равна среднему времени попадания в вершину y из вершины x (H_{xy}):*

$$\phi_{xy} = H_{xy} \tag{2.1}$$

Доказательство. Выберем начальный узел x и конечный узел y. В соответствии со вторым законом Кирхгофа, законом Ома и тем фактом, что все сопротивления проводников равны единице, имеем следующее:

$$d(x) = \sum_{z}^{\sim x} (\phi_{xy} - \phi_{zy}) \tag{2.2}$$

Разложим сумму разности на разность сумм. Заметим, что ϕ_{xy} не зависит от индекса z. Следовательно, суммирование ϕ_{xy} будет производиться столько раз, какую степень имеет вершина x. Тогда,(2.2) можно переписать:

$$d(x) = d(x) \cdot \phi_{xy} - \sum_{z}^{\sim x} \phi_{zy} \tag{2.3}$$

Решая уравнение, относительно ϕ_{xy}, получим следующее:

$$\phi_{xy} = 1 + \frac{1}{d(x)} \sum_{z}^{\sim x} \phi_{zy} \tag{2.4}$$

Введем дополнительное условие для случая $x = y$:

$$\phi_{xy} = \begin{cases} 0 & x = y \\ 1 + \frac{1}{d(x)} \sum\limits_{z}^{\sim x} \phi_{zy} & x \neq y \end{cases} \tag{2.5}$$

Из курса теории вероятности очевидно, что среднее время попадания из вершины x в вершину y (H_{xy}) равно среднему времени попадания из смежных с x вершин до вершины y плюс один (один шаг до смежной вершины).

$$H_{xy} = 1 + \frac{1}{d(x)} \sum_{z}^{\sim x} H_{zy} \tag{2.6}$$

Установим условие для $x = y$

$$H_{xy} = \begin{cases} 0 & x = y \\ 1 + \frac{1}{d(x)} \sum_{z}^{\sim x} H_{zy} & x \neq y \end{cases} \tag{2.7}$$

Заметим, что формулы (2.5) и (2.7) идентичны, т.к. являются линейными системами с единственным решением. Следовательно, получим:

$$H_{xy} = \phi_{xy} \tag{2.8}$$

□

Введём следующие определения.

Определение 2.0.5 (Сценарий B)**.** Сценарий B является схожим со сценарием A, за исключением того, что стоком будет являться вершина x. Обозначим разность потенциалов сценария B за ϕ'_{yx}, где x,$y \in V(G)$.

Используя теорему о времени первого попадания, сценарий B можно записать, как

$$\phi'_{yx} = H_{yx}$$

Определение 2.0.6 (Сценарий C)**.** Подадим ток, равный $2m$ на узел x, где m - количество связей цепи. Стоком будут являться все узлы сети. Обозначим разность потенциалов сценария C за ϕ''_{xy}, где x,$y \in V(G)$.

Данный сценарий является обратным к сценарию B, т.к. изменение затронуло направление тока в сети. Поэтому мы получили:

$$\phi''_{xy} = \phi'_{yx} = H_{yx}$$

Основываясь на этих определениях и теореме о времени первого попадания докажем следующую теорему.

Теорема 2.0.7 (Время возвращения)**.** *Время возвращения из вершины x через вершину y графа G равно удвоенному количеству ребер (m) графа G, умноженному на сопротивление R_{xy}[3].*

$$C_{xy} = 2mR_{xy} \tag{2.9}$$

Доказательство. Для доказательства этой теоремы используем сумму сценариев A и C. Другими словами, подадим ток $2m$ на вершину x и получим его с вершины y. Обозначим разность потенциалов в этом случае за ϕ'''_{xy}

$$\phi'''_{xy} = \phi_{xy} + \phi''_{yx}$$

По теореме о времени попадания и исходя из определения сценариев A и C получим:

$$\phi'''_{xy} = H_{xy} + H_{yx}$$

Заметим, что ϕ'''_{xy} является разностью потенциалов для перемещения тока $2m$ из узла x в узел y. Используя закон Ома имеем:

$$\phi'''_{xy} = 2mR_{xy}.$$

Тогда получаем

$$H_{xy} + H_{yx} = 2mR_{xy} \Rightarrow C_{xy} = 2mR_{xy} \tag{2.10}$$

□

3 Программа для расчета резистивного расстояния с помощью законов Кирхгофа

Для разработки программы был выбран язык программирования Octave[4], специально предназначенный для математических вычислений и являющейся бесплатной альтернативой MATLAB[5].

Основные задачи заключались в следующем:

а) Обеспечение универсальности (выбор файла с матрицей смежности графа, а также начального и конечного узлов).

б) Расчет резистивного расстояния между двумя произвольными вершинами графа.

в) Измерение времени выполнения программы.

Скрипт `mainKirg` на языке Octave, приведенный ниже.

⟨ *mainKirg* 19a ⟩ ≡

⟨ *start-up: input and prelimnary data check* 20 ⟩
⟨ *compute main part* 21 ⟩
⟨ *compute resistance-distatnce* 22a ⟩
⟨ *display results* 22b ⟩
◇

Fragment referenced in 19b.

"mainKirg.m" 19b≡
⟨ *mainKirg* 19a ⟩◇

3.1 Ввод и предварительная проверка данных

Начальный фрагмент определяет исходные данные задачи. Одним из ключевых аспектов являлась универсальность использования, что требовало реализации ввода имени файла для чтения и номеров узлов.

Здесь проверяется количество переданных параметров. В случае если их недостаточно, программа завершает работу с сообщением об ошибке. Далее происходит парсинг параметров: первый — имя файла, второй — начальный узел, третий — конечный узел. После происходит проверка введенных параметров на корректность. В случае неправильных данных вызывается ошибка с пояснением. Запускается таймер выполнения программы. В случае, если переданный файл уже обрабатывался, данные загрузятся из сохранённого файла. Иначе выполняется блок кода "compute main part".

⟨ *start-up: input and prelimnary data check* 20 ⟩ ≡

```
args = argv();
if numel(args) < 3
        error('Необходимо указать имя файла с матрицей смежности и',
                'номера двух узлов.');
end
filename = args{1};

a = str2double(args{2});
b = str2double(args{3});
if isnan(a) || isnan(b) || a <= 0 || b <= 0
        error('Номера узлов должны быть положительными целыми ',
                'числами.');
endif

tic;
[~, name, ~] = fileparts(filename);
pseudo_inverse_filename = [name, '_L_plus.mat'];

if exist(pseudo_inverse_filename, 'file') == 2
        load(pseudo_inverse_filename, 'L_plus');
        disp(['Псевдообратная матрица Лапласа загружена из файла ',
                pseudo_inverse_filename, '.']);
        n = size(L_plus, 1);
```

◇

Fragment referenced in 19a.

3.2 Основные расчеты

Этот блок отвечает за тяжеловесные расчеты программы.

Считывается матрица смежности из указанного файла. Подсчитывается количество вершин. Вычисляются степени вершин, и создается диагональная матрица степеней.

Затем вычисляется матрица Лапласиана. Происходит проверка корректности вычисления псевдообратной матрицы. Вычисляется псевдообратная матрица Лапласиана. Результат вычислений записывается в специальный файл. Считывание псевдообратной матрицы из этого файла позволит сократить время расчёта при очередном запуске программы с той же матрицей смежности.

⟨ *compute main part* 21 ⟩ ≡

```
else
        A = dlmread(filename);
        n = size(A, 1);


        D = diag(sum(A, 2));
        L = D - A;
        if rank(L) < n-1
                error('Матрица Лапласиана не имеет полного ранга.');
        end
        L_plus = pinv(L);
        save(pseudo_inverse_filename, 'L_plus');
        disp(['Псевдообратная матрица Лапласа вычислена и ',
                'сохранена в файл ', pseudo_inverse_filename, '.']);
end
```

◇

Fragment referenced in 19a.

3.3 Расчет разницы потенциалов

Проверяется, что указанные узлы существуют в матрице.

Создается результирующий вектор h и производится расчет резистивного расстояния. Для оптимизации программы эти действия записаны в одну строчку.

⟨ *compute resistance-distatnce* 22a ⟩ ≡

```
if a > n || b > n
        error('Номера узлов выходят за пределы размерности ',
                'матрицы.');
endif

answer = (L_plus(a, a) - L_plus(b, a)) - (L_plus(a, b) -
        L_plus(b, b));
```

◇

Fragment referenced in 19a.

3.4 Вывод результата

Остановка таймера выполнения и вывод результата:

⟨ *display results* 22b ⟩ ≡

```
elapsed_time = toc;
disp("============================================================");
disp(['Резистивное расстояние между узлами ', num2str(a), ' и ',
         num2str(b), ': ', num2str(answer)]);
disp(['Время выполнения программы: ', num2str(elapsed_time),
        ' секунд']);
```

◇

Fragment referenced in 19a.

4 Тестирование программы

Программа была протестирована на различных графах в три этапа:

а) Проверка правильности работы алгоритма на малых графах (до 5 вершин, заполненность 100%).

б) Тестирование на графах средних размеров (100-600 вершин, заполненность 90%).

в) Тестирование на больших графах (1000 и более вершин, заполненность 25-90%).

Данные для тестирования использовались без подгрузки предрасчитанной псевдообратной матрицы.

4.1 Результаты тестирования

— Малые графы: алгоритм выдает решение менее чем за сотую долю секунды.

— Средние графы: время выполнения менее полусекунды для графов с количеством вершин менее 500. При 500-600 вершинах расчет занимает 3.5 секунды.

— Большие графы: для графа с 1000 вершинами и заполненностью 90% расчет занимает 6.5 секунд. Граф с 3000 вершин и заполненностью 25% рассчитывается за 500 секунд. Больше данных изображено на Рис. 4.1

Основное время выполнения программы приходится на вычисление псевдообратной матрицы (сложность $O(n^3)$). Поэтому, время выполнения увеличивается на три порядка при увеличении размера графа в 10 раз.

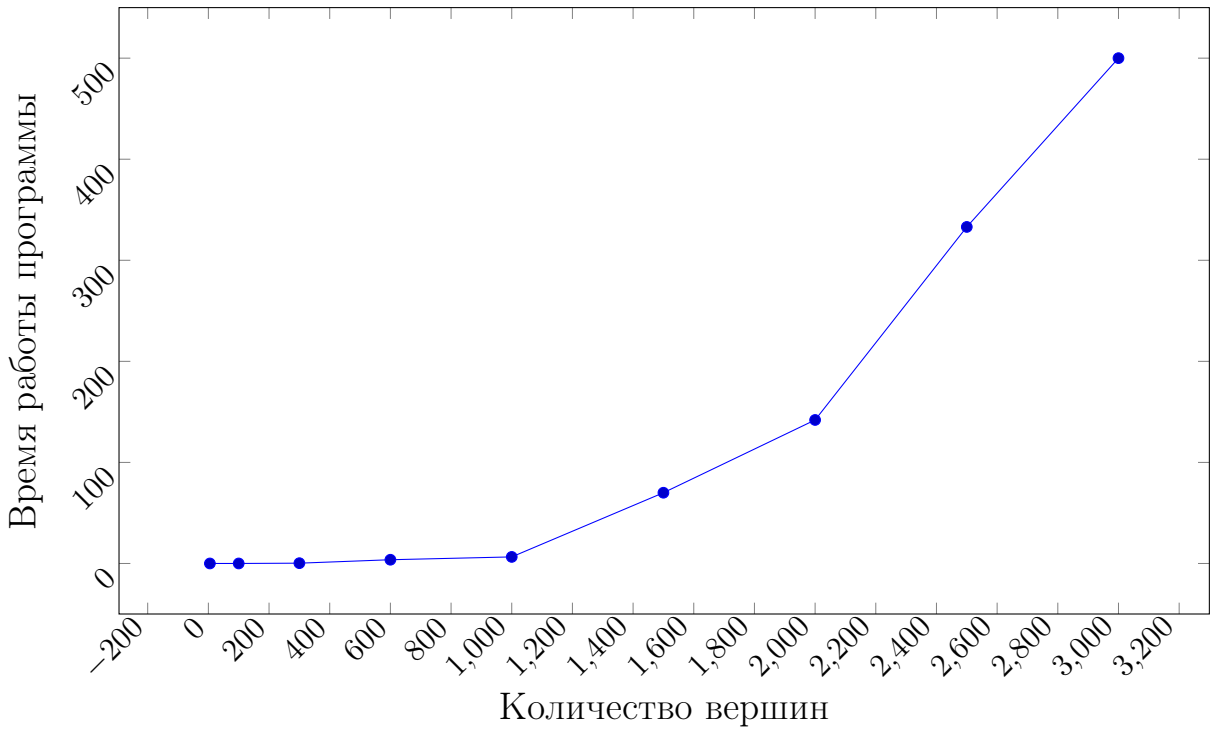

Рисунок 4.1 — Зависимость времени выполнения программы от количества вершин в графе

5 Реализация программы случайного блуждания

5.1 Переменные

5.1.1 Входные переменные

— `adj` - матрица смежности графа.

— `start` - индекс начального узла.

— `end_` - индекс конечного узла.

— `num_sim` - количество тестов.

5.1.2 Выходные переменные

— `fht` - вектор, содержащий время первого попадания в конечный узел для каждого теста.

— `ct` - вектор, содержащий время полного обхода графа для каждого теста.

— `cmt` - вектор, содержащий время перемещения от начального узла к конечному и обратно для каждого теста.

— `mfht` - среднее время первого попадания в конечный узел.

— `mct` - среднее время полного обхода графа.

— `mcmt` - среднее время перемещения от начального узла к конечному и обратно.

— `eff_res` - сопротивление графа, рассчитанное как среднее время перемещения, деленное на удвоенное количество ребер.

5.1.3 Промежуточные переменные

— `n` - количество узлов в графе.

— `m` - количество ребер в графе, рассчитанное как половина суммы всех элементов матрицы смежности.

— `curr` - текущий узел в процессе блуждания.

— `visited` - булевый вектор, указывающий, были ли посещены узлы.

— `steps` - количество шагов, сделанных в текущем тесте.

— `first_hit` - булева переменная, указывающая, было ли достигнуто первое попадание в конечный узел.

— `visit_count` - количество посещенных узлов.

— `neighbors` - вектор соседних узлов для текущего узла.

— `next` - следующий узел для перехода.

— `cms` - количество шагов для перемещения от начального узла к конечному и обратно в текущем тесте.

— `file_path` - путь к файлу с матрицей смежности.

— `elapsed_time` - время, затраченное на выполнение тестов.

5.2 Скрипт random walk.m

В данной работе была разработана программа на языке Octave, которая имитирует случайные блуждания по графу, заданному матрицей смежности. Программа рассчитывает статистические данные. Об этих данных ниже рассказано более подробно.

⟨ *random walk* 27a ⟩ ≡

```
function [fht, ct, mfht, mct, eff_res, ...
mcmt] = random_walk(adj, start, end_, num_sim)
⟨ initialization of variables 27c ⟩
⟨ simulation loop 28a ⟩
⟨ calculation 31a ⟩
◇
```

Fragment referenced in 27b.

"random_walk.m" 27b≡

⟨ *random walk* 27a ⟩◇

В этом блоке инициализируются переменные для количества узлов и ребер в графе, векторов для хранения времени первого попадания, времени полного обхода, и времени перемещения.

⟨ *initialization of variables* 27c ⟩ ≡

```
n = size(adj, 1);
m = sum(adj(:)) / 2;
fht = zeros(num_sim, 1);
ct = zeros(num_sim, 1);
cmt = zeros(num_sim, 1);
◇
```

Fragment referenced in 27a.

Этот блок выполняет цикл по числу тестов для выполнения случайных блужданий.

⟨ *simulation loop* 28a ⟩ ≡

⟨ *initialising the current state* 28b ⟩
⟨ *graph wandering cycle* 29a ⟩
⟨ *hit check* 29b ⟩
⟨ *update visited nodes* 30a ⟩
⟨ *record round-trip time ?* ⟩
⟨ *calculation of travel time* 30b ⟩
◇

Fragment referenced in 27a.

Этот блок начинает наш основной цикл, инициализируя текущий узел как начальный узел , а также массивы для отслеживания посещенных узлов и количества шагов. Также инициализируются переменные для отслеживания первого попадания в конечный узел и количества посещенных узлов.

⟨ *initialising the current state* 28b ⟩ ≡

```
for sim = 1:num_sim
    curr = start;
    visited = false(n, 1);
    visited(start) = true;
    steps = 0;
    first_hit = false;
    visit_count = 1;
```

◇

Fragment referenced in 28a.

Этот цикл продолжается до тех пор, пока не будут посещены все узлы графа. Внутри цикла определяется следующий узел для перехода случайным образом из соседей текущего узла.

⟨ *graph wandering cycle* 29a ⟩ ≡

```
while visit_count < n
    neighbors = find(adj(curr, :));
    next = neighbors(randi(length(neighbors)));
    steps = steps + 1;
    curr = next;
```

◇

Fragment referenced in 28a.

Если это первое попадание в конечный узел, сохраняется количество шагов до первого попадания.

⟨ *hit check* 29b ⟩ ≡

```
if ~first_hit && curr == end_
    fht(sim) = steps;
    first_hit = true;
end
```

◇

Fragment referenced in 28a.

Если узел еще не был посещен, он помечается как посещенный, и увеличивается счетчик посещений. После завершения обхода всех узлов записывается количество шагов.

⟨ *update visited nodes* 30a ⟩ ≡

```
            if ~visited(curr)
                visited(curr) = true;
                visit_count = visit_count + 1;
            end
        end

        ct(sim) = steps;
```

◇

Fragment referenced in 28a.

Затем рассчитывается время перемещения от начального узла к конечному и обратно. Это выполняется двумя циклами: один до конечного узла, и один обратно к начальному узлу.

⟨ *calculation of travel time* 30b ⟩ ≡

```
        cms = 0;
        curr = start;
        while curr ~= end_
            neighbors = find(adj(curr, :));
            next = neighbors(randi(length(neighbors)));
            cms = cms + 1;
            curr = next;
        end
        while curr ~= start
            neighbors = find(adj(curr, :));
            next = neighbors(randi(length(neighbors)));
            cms = cms + 1;
            curr = next;
        end
        cmt(sim) = cms;
    end
```

◇

Fragment referenced in 28a.

Здесь рассчитывается среднее время первого попадания, среднее время обхода, среднее время перемещения и сопротивление.

⟨ *calculation* 31a ⟩ ≡

```
    mfht = mean(fht);
    mct = mean(ct);
    mcmt = mean(cmt);

    eff_res = mcmt / (2 * m);
end
```

◇

Fragment referenced in 27a.

5.3 Скрипт run random walk.m

Для запуска функции random walk и получения результатов был разработан скрипт run random walk. Этот скрипт задаёт параметры графа, запускает функцию random walk и выводит результаты на экран.

⟨ *run random walk* 31b ⟩ ≡

⟨ *parameters set-up* 32a ⟩
⟨ *running simulation* 32b ⟩
⟨ *main results output* 33 ⟩
◇

Fragment referenced in 31c.

"run_random_walk.m" 31c≡

⟨ *run random walk* 31b ⟩◇

Указание пути к файлу с матрицей смежности, начального узла, конечного узла и числа тестов. Затем чтение матрицы смежности из указанного

файла с помощью функции `dlmread` и сохранение в переменную матрицы смежности.

⟨ *parameters set-up* 32a ⟩ ≡

```
args = argv();
if numel(args) < 3
        error('Необходимо указать имя файла с матрицей смежности и',
                'номера двух узлов.');
end

file_path = args{1};
start = str2double(args{2});
end_ = str2double(args{3});
num_sim = 1000;

adj = dlmread(file_path);
```

◇

Fragment referenced in 31b.

Выполнение функции случайного блуждания с заданными параметрами и измерение затраченного времени с помощью `tic` и `toc`. Результаты сохраняются в соответствующих переменных, а время выполнения - в переменной `elapsed_time`.

⟨ *running simulation* 32b ⟩ ≡

```
tic;
[fht, ct, mfht, mct, eff_res, mcmt] = random_walk(adj, start, end_, num_sim);
elapsed_time = toc;
```

◇

Fragment referenced in 31b.

Этот блок кода выводит основные результаты вычислений, такие как: среднее время первого попадания, среднее время прохода, среднее время обхода всего графа, сопротивление, время выполнения программы

⟨ *main results output* 33 ⟩ ≡

```
fprintf('Среднее время первого попадания в вершину %d из' ,
        'вершины %d: %f шага.\n', end_, start, mfht);
fprintf('Среднее время прохода из вершины %d в вершину %d и' ,
        'обратно: %f шага.\n', start, end_, mcmt);
fprintf('Среднее время обхода всего графа: %f шага.\n', mct);
fprintf('Сопротивление: %f.\n', eff_res);
fprintf('Время выполнения программы: %f секунд.\n', elapsed_time);
```

◇

Fragment referenced in 31b.

6 Результаты и их анализ

6.1 Проведение тестов

В качестве примеров мы рассмотрим три графа. Первый граф будет тестовым для понимания методологии решения, вторым графом является Московское метро, третий – линейный граф.

6.1.1 Пример 1

$$\text{adj\_matrix} = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

Начальная вершина была выбрана как вершина 1, а конечная вершина как вершина 4. Количество тестов было установлено на 1000. Результаты тестов показали следующее:

— Среднее время первого попадания в вершину 4 из вершины 1: 5.047200 шага.

— Среднее время прохода из вершины 1 в вершину 4 и обратно: 10.004400 шага.

— Среднее время обхода всего графа: 5.934400 шага.

— Сопротивление: 1.000440.

— Время выполнения программы: 7.940940 секунд.

Стоит заметить, что при стократном увеличении тестов, результаты изменятся незначительно.

6.1.2 Пример 2

В качестве второго примера была взята матрица Московского метро (442 вершины, 630 ребер), так как она позволяет проверить случайное

блуждание в реальной транспортной сети. Начальная вершина была выбрана как вершина 1(станция Новокосино), а конечная вершина как вершина 442(станция Апрелевка). Количество тестов было установлено на 100000. Результаты тестов показали следующее:

— Среднее время первого попадания в вершину 442 из вершины 1: 17159.75 шага.

— Среднее время прохода из вершины 1 в вершину 442 и обратно: 20142.68 шага.

— Среднее время обхода всего графа: 40777.67 шага.

— Сопротивление: 15.99.

— Время выполнения программы: 3194.03 секунд.

Резистивное расстояние по результатам выполнения программы для расчёта сопротивления с помощью законов Киргхофа равно 16. Тогда из формулы (2.9) следует, что среднее время коммутирования равно:

$$C^t_{(1,442)} = 2mR_{1,442} = 2 * 630 * 16 = 20160$$

Среднее время коммутирования по результатам случайных блужданий:

$$C^r_{(1,442)} \approx 20142$$

Результаты вычислений показывают, что среднее время прохождения из вершины 1 в вершину 442 и обратно почти точно удовлетворяет соотношению формулы (2.9).

$$C^t_{(1,442)} \approx C^r_{(1,442)}$$

Ошибка составлятет 0,01%.

6.1.3 Пример 3

Третьим примером стал линейный граф с количеством вершин 442 и количеством ребер 441. Количество тестов было установлено на 10000.

— Среднее время первого попадания в вершину 442 из вершины 1: 191980.58 шага.

— Среднее время прохода из вершины 1 в вершину 442 и обратно: 389058.45 шага.

— Среднее время обхода всего графа: 191980.58.

— Сопротивление: 441.11.

— Время выполнения программы: 32729.89 секунд.

Резистивное расстояние по результатам выполнения программы для расчёта сопротивления с помощью законов Киргхофа равно 441. Тогда из формулы (2.9) следует, что среднее время коммутирования равно:

$$C^t_{(1,442)} = 2mR_{1,442} = 2 * 441 * 441 = 388962$$

Среднее время коммутирования по результатам случайных блужданий:

$$C^r_{(1,442)} \approx 389058.45$$

Результаты вычислений показывают, что среднее время прохождения из вершины 1 в вершину 442 и обратно достаточно точно удовлетворяет соотношению формулы (2.9).

$$C^t_{(1,442)} \approx C^r_{(1,442)}$$

Ошибка составлятет 0,03%.

6.2 Теоретические расчеты

6.2.1 Пример 1

Теперь с помощью теории рассчитаем количество шагов до попадания в вершину 4 из вершины 1. Составим матрицу степеней из матрицы смежности, используемой выше:

$$\mathrm{D} = \begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 2 \end{pmatrix}$$

Из этих двух матриц составим матрицу Лапласа (Лапласиан), используя данную формулу: L = D - adj_matrix

$$\begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 2 \end{pmatrix} - \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 2 & -1 & -1 & 0 \\ -1 & 3 & -1 & -1 \\ -1 & -1 & 3 & -1 \\ 0 & -1 & -1 & 2 \end{pmatrix} = \mathrm{L}$$

Далее мы получаем собственные значения и собственные вектора матрицы Лапласа, чтобы их получить я воспользуюсь Octave:

⟨ *laplacian* 37a ⟩ ≡

```
laplacian_matrix = [2 -1 -1 0;
                    -1 3 -1 -1;
                    -1 -1 3 -1;
                     0 -1 -1 2];


[eigenvectors, eigenvalues_matrix] = eig(laplacian_matrix);

eigenvalues = diag(eigenvalues_matrix);

disp('Собственные значения:');
disp(eigenvalues);

disp('Собственные вектора:');
disp(eigenvectors);
```

◇

Fragment referenced in 37b.

"laplacian.m" 37b≡

⟨ *laplacian* 37a ⟩◇

Где eigenvalues – это λ_k, а eigenvectors – это u_i

$$\lambda_1 = 0, \quad \lambda_2 = 2, \quad \lambda_3 = 4, \quad \lambda_4 = 4$$

$$\mathrm{U} = \begin{pmatrix} -0.50 & -0.71 & 0.49 & 0.09 \\ -0.50 & 0.00 & -0.62 & 0.60 \\ -0.50 & 0.00 & -0.36 & -0.79 \\ -0.50 & 0.71 & 0.49 & 0.09 \end{pmatrix}$$

У собственных значений убираем нулевое значения и берем нулевую и третью строки из матрицы U для расчета R_{14}. Получаем следующее:

$$\lambda_2 = 2, \quad \lambda_3 = 4, \quad \lambda_4 = 4$$

$$u_0 = (-0.50, -0.71, 0.49, 0.09)$$

$$u_3 = (-0.50, 0.71, 0.49, 0.09)$$

Теперь нам нужны значения n и m. У нас n уже известна, она равна 4, поэтому осталось найти m. Для этого находим сумму всех элементов матрицы смежности и делим их на 2.

$$\sum_{i,j} A_{ij} = 0+1+1+0+1+0+1+1+1+1+0+1+0+1+1+0 = 10$$

$$E = \frac{10}{2} = 5$$

В итоге получаем, что $m = 5$ и $n = 4$. Получив эти значения, мы можем посчитать сопротивление

$$R_{14} = \sum_{k=1}^{n-1} \frac{(u_{k1} - u_{k4})^2}{\lambda_k} = \frac{(0.71 - (-0.71))^2}{2} +$$

$$+ \frac{(0.49 - 0.49)^2}{4} + \frac{(0.09 - 0.09)^2}{4} \approx 1.0082$$

Теперь нужно посчитать commute time, для этого воспользуемся формулой (2.10):

$$C_{14} = 2mR_{14} = 2 * 5 * 1.0083 \approx 10.0082$$

Нам осталось посчитать только hitting time, но так как наш граф является маленьким и симметричным, то у нас есть возможность воспользоваться данным вариантом формулы (2.10):

$$C_{14} = H_{14} + H_{41}$$

Для нашего маленького и симметричного графа, это равносильно что:

$$C_{14} \approx 2H_{14}$$

Следовательно H_{14} равен:

$$H_{14} \approx \frac{C_{14}}{2} \approx \frac{10.0082}{2} \approx 5.0041$$

6.2.2 Пример 2

Пример 1 был приведен для демонстрации методологии. Поскольку нахождение вручную данных графа Московского метро практически невозможно, воспользуемся программным методом. Для этого была написана программа, которая воспроизводит вычисления формул: R_{xy}, H_{xy}, C_{xy}.

⟨ *theoretical random walk* 39 ⟩ ≡

⟨ *parameters* 40b ⟩
⟨ *calculation of eigenvalues and eigenvectors* 41a ⟩
⟨ *parameters of the iterative process* 41b ⟩
⟨ *neighbourhood lists* 42 ⟩
⟨ *calculate the matrix H* 43 ⟩
⟨ *calculation of results* 44a ⟩
⟨ *result output* 44b ⟩
◇

Fragment referenced in 40a.

"theoretical_random_walk.m" 40a≡

⟨ *theoretical random walk* 39 ⟩◇

В этой части кода, мы загружаем матрицу смежности графа из файла. Затем определяем количество вершин и ребер.

⟨ *parameters* 40b ⟩ ≡

```
args = argv();
if numel(args) < 3
        error('Необходимо указать имя файла с матрицей смежности и',
                'номера двух узлов.');
end
file_path = args{1};
adj_matrix = dlmread(file_path);

n = size(adj_matrix, 1);
m = sum(adj_matrix(:)) / 2;
```

◇

Fragment referenced in 39.

Здесь мы вычисляем по формулам собственные значения и собственные векторы. Далее извлекаем собственные значения в вектор. И отбрасываем первое нулевое собственное значение и соответствующий ему собственный вектор (они не нужны для дальнейших вычислений).

⟨ *calculation of eigenvalues and eigenvectors* 41a ⟩ ≡

```
degree_matrix = sum(adj_matrix, 2);

laplacian_matrix = diag(degree_matrix) - adj_matrix;

[eigenvectors, eigenvalues_matrix] = eig(laplacian_matrix);
eigenvalues = diag(eigenvalues_matrix);

eigenvalues_nonzero = eigenvalues(2:end);
eigenvectors_nonzero = eigenvectors(:, 2:end);

◇
```

Fragment referenced in 39.

В этой части код инициализируем матрицу H нулями. Задаем максимальное количество итераций и допустимую ошибку.

⟨ *parameters of the iterative process* 41b ⟩ ≡

```
H = zeros(n, n);

max_iter = 200000;
tol = 1e-16;
◇
```

Fragment referenced in 39.

Здесь начинаем подсчет времени выполнения. Затем создаем список соседей для каждой вершины. В каждом элементе этого массива neighbors будут храниться массивы индексов соседей соответствующей вершины.

⟨ *neighbourhood lists* 42 ⟩ ≡

```
tic;

neighbors = cell(n, 1);
for i = 1:n
    neighbors{i} = find(adj_matrix(i, :) == 1);
end
```

◇

Fragment referenced in 39.

В этом блоке в каждой итерации обновляем значения H для всех вершин, используя их соседей. Затем каждые 10000 итераций выводим текущую ошибку. Если ошибка становится меньше заданного порога (tol), процесс останавливается. И в конце записываем время выполнения итерационного процесса.

⟨ *calculate the matrix H* 43 ⟩ ≡

```
for iteration = 1:max_iter
    H_prev = H;
    for u = 1:n
        if degree_matrix(u) > 0
            H(u, :) = 1 + (1 / degree_matrix(u)) * sum(H(neighbors{u}, :), 1)
        end
        H(u, u) = 0;  % Условие H_ii = 0
    end
    if mod(iteration, 10000) == 0
        disp(['Итерация: ', num2str(iteration), ' H - ', num2str(max(max(abs(H
    end
    if max(max(abs(H - H_prev))) < tol
        disp(['Сошлось после ', num2str(iteration), ' итераций']);
        break;
    end
end
elapsed_time = toc;
```

◇

Fragment referenced in 39.

Здесь мы рассчитаем среднее время прохождения, время возвращения и сопротивление между заданными вершинами на основе полученных данных.

⟨ *calculation of results* 44a ⟩ ≡

```
i = str2double(args{2});
j = str2double(args{3});

H_ij = H(i, j);

R_ij = sum((eigenvectors_nonzero(i, :) - eigenvectors_nonzero(j, :)).^2 ./ eig

C_ij = 2 * m * R_ij;
```
◇

Fragment referenced in 39.

Выводим вычисленные результаты.

⟨ *result output* 44b ⟩ ≡

```
disp(['Среднее время прохода из вершины ', num2str(i), ' в вершину ', num2str
disp(['Среднее время прохода из вершины ', num2str(i), ' в вершину ', num2str
disp(['Сопротивление: ', num2str(R_ij)]);
disp(['Время выполнения программы: ', num2str(elapsed_time), ' секунд']);
```
◇

Fragment referenced in 39.

Результаты данных вычислений:

— Сошлось после 293492 итераций

— Среднее время прохождения из вершин 1 в вершину 442: 17085.752085552034

— Среднее время прохождения из вершины 1 в вершину 442 и обратно: 20199.377213668093

— Сопротивление: 16.03125175687944

— Время выполнения программы: 2841.8106 секунд

6.2.3 Пример 3

Результаты вычислений для линейного графа:

— Сошлось после 2016601 итераций

— Среднее время прохождения из вершин 1 в вершину 442: 194480.99999830942

— Среднее время прохождения из вершины 1 в вершину 442 и обратно: 388961.9999977657

— Сопротивление: 440.9999999974668

— Время выполнения программы: 8452.6199 секунд

Результаты вычислительных экспериментов и теоретических расчетов для графа демонстрируют высокую степень согласованности, что свидетельствует о правильности применяемых моделей и алгоритмов. Тестирование проводилось на графе с четырьмя вершинами, на котором было установлено, что теоретические предсказания совпадают с практическими результатами. Кроме того, применение данных методов на лиенйном графе и на графе Московского метро показало, что алгоритмы могут работать не только в теоретических условиях, но и с реальной транспортной сетью. Среднее время первого попадания, прохождения и обхода всего графа, а также сопротивление, полученные в результате тестов, находятся в хорошем согласии с теоретическими расчетами. Эти три случая подтверждают, что предложенные модели и алгоритмы могут быть успешно применены для анализа и прогнозирования поведения более сложных и масштабных транспортных сетей.

ЗАКЛЮЧЕНИЕ

В рамках данной исследовательской работы была рассмотрена задача нахождения резистивного расстояния между узлами электрической цепи, представленной в виде графа. В результате проведенных теоретических исследований и разработок удалось достичь следующих результатов:

а) **Математическое моделирование электрических цепей**:

— Электрические цепи были успешно смоделированы с помощью теории графов, где узлы сети представляли вершины графа, а сопротивления между узлами — веса ребер.

— Была установлена взаимосвязь между Законами Кирхгофа и резистивным расстоянием, что позволяет использовать математические методы теории графов для анализа электрических цепей.

б) **Вывод резистивного расстояния**:

— Резистивное расстояние между двумя вершинами графа было определено как сопротивление между соответствующими узлами в электрической цепи, где все сопротивления заменены на единичные.

— Была выведена формула для нахождения резистивного расстояния с использованием обобщенной обратной матрицы Лапласиана графа, что позволило упростить вычисления и сделать их более наглядными.

в) **Разработка программ**:

— На языке Octave были разработаны алгоритмы для вычисления резистивного расстояния между двумя вершинами графа, а также алгоритм случайного блуждания. Программы обеспечивают универсальность использования, позволяют загружать матрицу смежности из файла и вычислять ффективное сопротивление и метрики случайных блужданий.

— Программа была протестирована, результаты тестирования показали ее корректность.

г) **Практическое применение**:

— Разработанные модели и программа могут быть использованы для анализа и оптимизации различных сетей, включая электрические, транспортные и коммуникационные сети.

— Методика может быть полезна в таких областях, как проектирование электрических цепей, анализ надёжности сетей и оптимизация маршрутов.

Таким образом, проделанная работа позволяет сделать вывод о том, что применение теории графов и метода случайных блужданий для анализа электрических цепей является перспективным направлением исследований.

Разработанные алгоритмы и программы могут быть полезны в практике инженеров и исследователей, занимающихся анализом и оптимизацией сложных сетей.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. *Klein, Douglas J.* Resistance-Distance Sum Rules / Douglas J. Klein. — Croatica Chemica Acta, 2002.

2. *Klein D. J., Randic M. J.* Resistance Distance / Randic M. J. Klein D. J. — J. Math. Chem, 1993.

3. The Electrical Resistance of a Graph Captures its Commute and Cover Times / Ashok K. Chandra, Prabhakar Raghavan, Walter L. Ruzzo et al. — 1989.

4. Octave - это высокоуровневый язык программирования и интерактивная вычислительная среда, предназначенная в первую очередь для решения численных задач. Он был разработан как свободная альтернатива коммерческому пакету MATLAB, поддерживая большинство его функциональных возможностей.

5. MATLAB (сокращение от "Matrix Laboratory") — это высокоуровневый язык программирования и интерактивная вычислительная среда, используемая главным образом для численных вычислений, визуализации данных и разработки алгоритмов.