Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ
Н.Г.ЧЕРНЫШЕВСКОГО»

Кафедра физики открытых систем

наименование кафедры

Индуцированные шумом переключения в сети фазовых осцилляторов

наименование темы выпускной квалификационной работы

Курамото. Разработка программного кода и численное моделирование

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 431 группы

направления 09.03.02 Информационные системы и технологии

код и наименование направления

Факультета нелинейных процессов

наименование факультета

Кириллова Олега Александровича

фамилия, имя, отчество

Научный руководитель:

профессор, д.ф.-м. наук, доцент		О.И. Москаленко
должность, уч. ст., уч. зв.	личная подпись, дата	инициалы, фамилия

Зав. кафедрой физики открытых систем:

профессор, д.ф.-м. наук, профессор		А.А.Короновский
должность, уч. ст., уч. зв.	личная подпись, дата	инициалы, фамилия

Саратов 2018 г.

Введение

Целью бакалаврской работы является разработка программного пакета для изучения синхронизации в сети фазовых осцилляторов Курамото и изучение влияния шума на установление синхронного режима в исследуемой сети на основе разработанного кода.

Синхронизация - универсальный феномен, играющий огромную роль во множестве процессов в разных областях науки. Изучением синхронизации занимаются многие современные ученые, так как понимание механики этого процесса помогает более подробно изучить поведение сложных систем, найти причины эпилепсии, каскадного отключения электрических систем и, даже, возникновения нелинейного поведения трафика в сети Интернет. Влияние шумов разного рода на распределенные системы неизбежно. Исследование этой области позволит разработать методы для более точного изучения данных систем.

Удобной и распространенной моделью для изучения синхронизации в сложных системах является сеть фазовых осцилляторов Курамото. Она позволяет моделировать большинство таких ситуаций путем незначительного изменения параметров и/или уравнений системы. К сожалению, для получения наглядных результатов, необходимо моделировать сеть с большим количеством узлов, что отрицательно сказывается на времени выполнения. Применение технологий параллельных вычислений также является приоритетной задачей.

В ходе работы было произведено сравнение технологий параллельных вычислений OpenCL и OpenMP, написана библиотека на языке программирования C++, предоставляющая гибкий и удобный интерфейс взаимодействия. Также был разработан интерфейс командной строки, написанный с использованием фреймворка Qt, позволяющий легко конфигурировать сеть и выполнять расчеты. В ходе численного моделирования было изучено влияние шума на классическую и адаптивную однослойные модели фазовых осцилляторов Курамото и получена зависимость ширины петли от интенсивности шума для адаптивной модели.

Бакалаврская работа состоит из 30 страниц текста, включая 9 иллюстраций, 2 таблицы и 6 листингов. Список источников содержит 8 наименований.

1 Сеть фазовых осцилляторов Курамото

Сеть фазовых осцилляторов Курамото — математическая модель, описывающая поведение большого количества осцилляторов, использующаяся для изучения явления синхронизации, предложена японским физиком Йосики Курамото (Kuramoto Yoshiki). Классическая модель имеет вид:

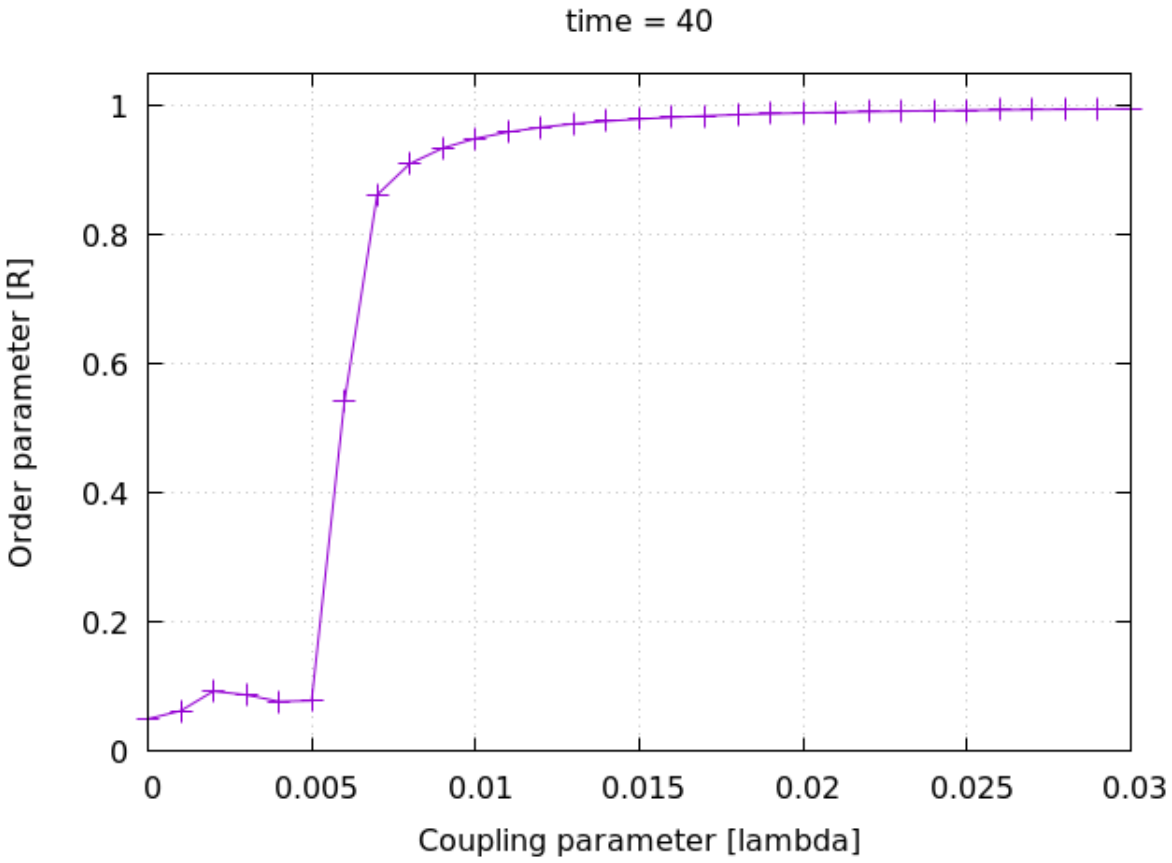

Рис. 1: Установление синхронизации в классической сети фазовых осцилляторов Курамото. $time = 40$, $\lambda \in [0.00; 0.03]$

В ходе исследований классической системы была открыта взрывная синхронизация (explosive synchronisation), которая появляется при условии некоторой корреляции между собственными частотами осцилляторов [1]. Для изучения взрывной синхронизации без присутствия корреляции в системе была разработана адаптивная модель:

В современном мире сложно найти систему, на которую бы не оказывалось влияние извне. Данное влияние можно классифицировать как шумовое, что открывает новые возможности для исследования привычных и изученных систем. Классическая и адаптивные системы в присутствии шума выглядят следующим образом:

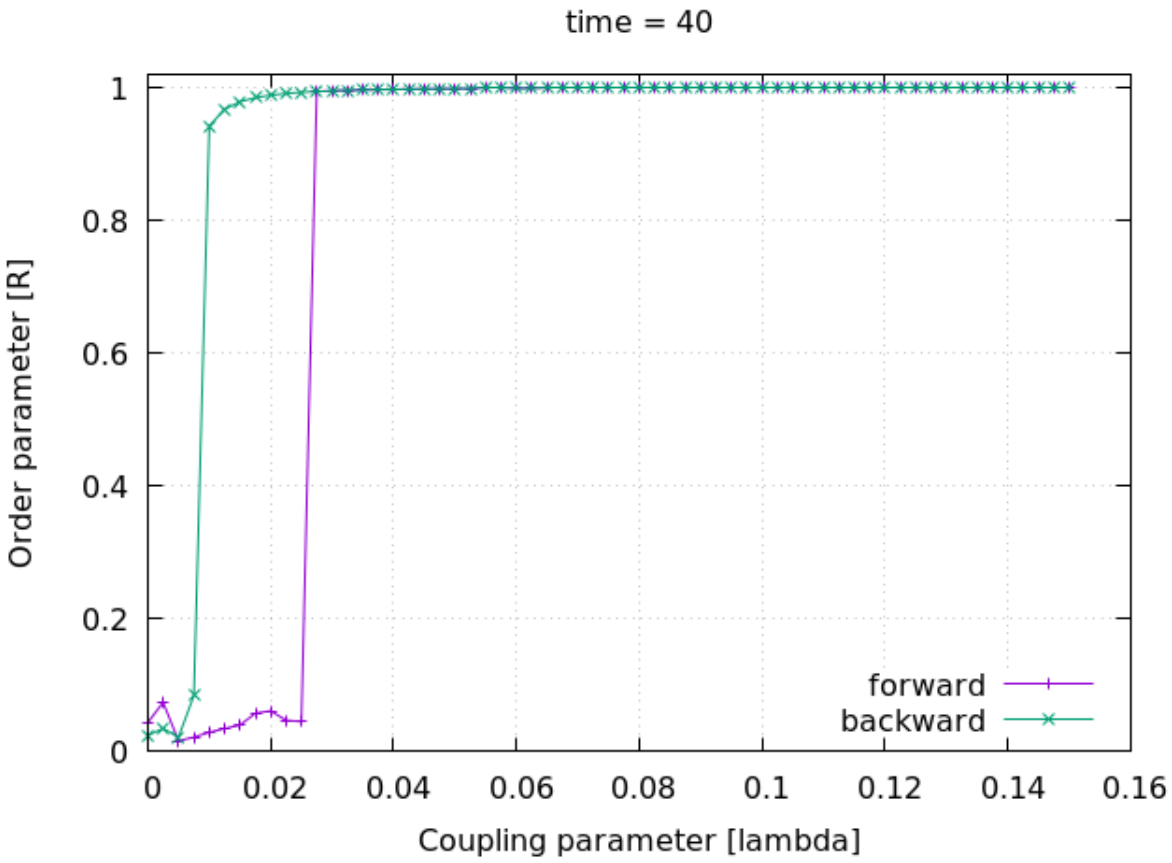

Рис. 2: Установление синхронизации в адаптивной сети фазовых осцилляторов Курамото для двух направлений изменения параметра связи. $time = 40$, $\lambda \in [0.00; 0.15]$

$$\dot{\theta}_i = D\xi + \omega_i + \lambda \sum_{j=1}^{N} A_{ij} sin(\theta_j - \theta_i); \tag{1}$$

$$\dot{\theta}_i = D\xi + \omega_i + \lambda\alpha_i \sum_{j=1}^{N} A_{ij} sin(\theta_j - \theta_i), \tag{2}$$

Изучение влияния шума на системы позволит разработать методы управления поведением систем, а, также, исследовать возможности стабилизации систем[1], в которых присутствует явление взрывной синхронизации, с помощью внешнего воздействия.

[1]Здесь и далее, под стабилизацией адаптивной системы понимается сокращение разницы между критическими значениями параметра связи λ, при которых происходит переключение режима.

2 Стек технологий

Для реализации программы был выбран язык программирования C++. Данный выбор обоснован как широкими возможностями данного языка программирования, так и высокой скоростью исполнения машинного кода. Также данный язык программирования имеет обширную стандартную библиотеку, что упрощает работу с многомерными массивами, комплексными числами и утилитарными алгоритмами, такими как нахождение минимума или максимума или суммирование элементов массива.

2.1 OpenMP

OpenMP — Open Multi-Processing — открытый программный пакет для параллелизации программного кода на языках C, C++ и Fortran. Данный пакет представляет собой набор директив препроцессора для генерации параллельного кода и переменных среды выполнения для поддержания работы программ. Данный метод распараллеливания, хоть и был выпущен в 1997 году, но до сих пор имеет регулярные обновления и интенсивно используется по всему миру.

Технология OpenMP проста в использовании, хотя и обладает внушительным функционалом. Это позволяет быстро ускорить существующий код, не тратя ресурсы на изучение сложных технологий (таких как pthreads). Но стоит отметить и минусы данной технологии: OpenMP не поддерживает вычисления на GPU[2]; OpenMP поддерживает очень ограниченный круг языков программирования (C, C++ и Fortran) и, хотя, в большинстве случаев этого вполне хватает, в последнее время активно развивается язык программирования Python, обладающий существенными преимуществами перед вышеперечисленными языками.

2.2 OpenCL

OpenCL — Open Computing Language — фреймворк для написания параллельного программного кода на различных устройствах, таких как цен-

[2]В условиях отсутствия суперкомпьютера или ЛВС с настроенной технологий MPI, вычисления на видеокарте, в большинстве случаев, показывают существенный прирост производительности по сравнению с технологией OpenMP.

тральные процессоры (CPU), графические процессоры (GPU), со-процессоры и программируемые вентильные матрицы (Accelerator). Данная технология была предложена Apple и AMD в начале 2008 года, и вскоре, было внесено предложение о разработке спецификации в консорциум Khronos Group. Стандарт OpenCL 1.0 был обнародован 9 июня 2008, но первый выпуск был вместе с выпуском Mac OS X 10.6 28 августа 2009 года. В комитет по стандартизации OpenCL вошли такие компании как Apple, AMD, IBM, nVidia, Intel и ARM.

OpenCL представляет собой библиотеку поддержки выполнения, индивидуальную для каждого вендора, и набор заголовочных файлов, реализующих интерфейс взаимодействия с устройством. Каждая программа с использованием OpenCL состоит из двух частей: управляющего кода(host) и ядра(kernel). Host отвечает за подготовку данных и устройства, компилирование kernel-кода, контролирование процесса выполнения и за вывод данных с устройства. Kernel - является мини-программой, написанной на языке программирования CL C[3], содержащий логику расчетов и вспомогательные функции. Также стоит отметить, что для OpenCL доступны т.н. нативные функции (native functions). Это специальным образом, часто аппаратно, оптимизированные функции для часто используемых действий при расчетах и/или обработке изображений. Данные функции в основном присутствуют таких типах OpenCL устройств, как GPU и ACCELERATOR. В тех устройствах, где они не поддерживается, вызываются стандартные аналоги.

2.3 Сравнение OpenMP и OpenCL

OpenMP подходит для быстрой модификации параллелизации уже готовых алгоритмов. Декларативность и низкий порог входа избавляют ученого от изучения дополнительных технологий при написании программного кода, что позволяет сконцентрироваться на самом алгоритме.

Хотя OpenCL требует значительной подготовки при написании программного кода, данная технология позволяет производить расчеты практически на любом оборудовании. Таким образом, данная технология избавляет ученого от необходимости в супер-компьютерах, позволяя использовать домашний или рабочий персональный компьютер в качестве расчетной машины. Также, стоит отметить, что при использовании OpenCL ученый может использовать

[3]Язык программирования CL C основан на стандарте языка программирования C 99.

одну и ту же программу как на супер-компьютере (запустив ее на CPU), так и на персональном компьютере, производя расчет на GPU, что существенно сократит время ожидания результатов.

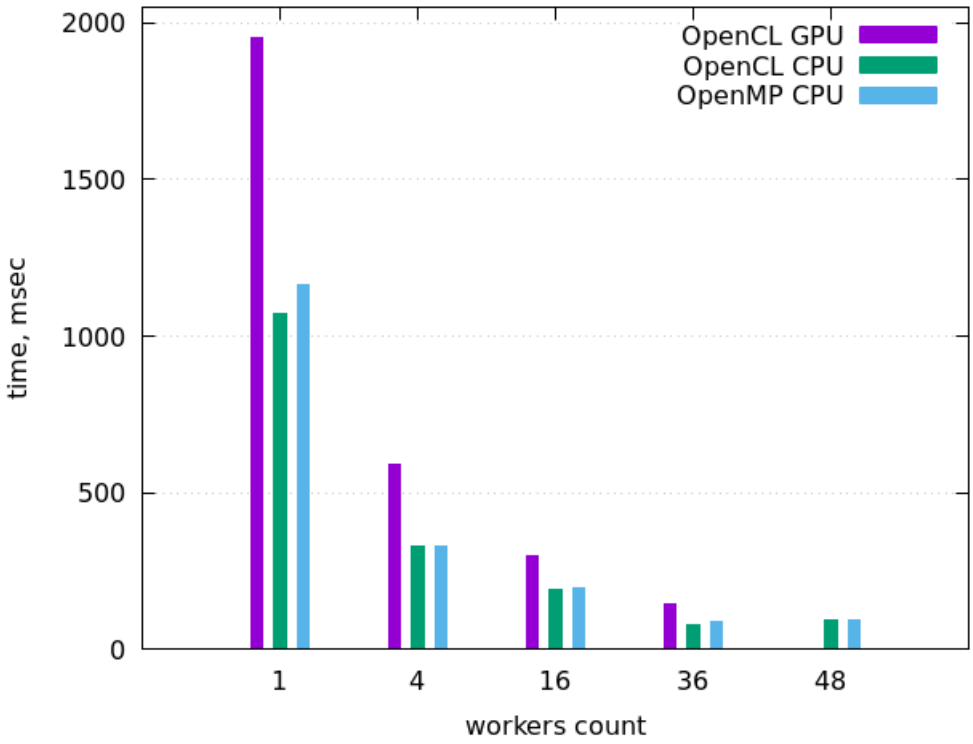

Рис. 3: Сравнение производительности OpenCL и OpenMP.

Обобщая вышесказанное, OpenMP подходит для быстрого распараллеливания уже готового программного кода, не требуя больших затрат от ученого. Программы с использованием OpenCL сложны в реализации и требуют изучения данной технологии, но в условиях отсутствия промышленной вычислительной техники, позволяют использовать персональные компьютеры для высокопроизводительных расчетов. Также стоит отметить, что промышленное вычислительное оборудование стоит дороже нескольких потребительских видеокарт, обеспечивающих схожую производительность.

3 Реализация программного кода

Как сказано выше, для реализации программного кода был выбран язык программирования C++. Выбор был обоснован удобством данного языка и скоростью выполнения. Была выбрана архитектура библиотеки, что позволит в дальнейшем более гибко использовать данный код. Также был разработан интерфейс командной строки для более удобного эксплуатирования программного кода.

3.1 Библиотека

Правильный выбор архитектуры для программы является ключевым этапом в начале разработки. От данного решения зависит удобство использования программного пакета, а, соответственно, его популярность.

Поставляется библиотека двумя файлами[4]: заголовочным и объектным, что позволяет программисту, вместо настройки окружения для компиляции библиотеки сосредоточиться на ее использовании, а также скрывает реализацию алгоритма в том случае, если код является проприетарным. Заголовочный файл включается в исходный код новой программы, а объектный подсоединяется компилятором.

3.2 Интерфейс командной строки

Для использования библиотеки был разработан интерфейс командной строки[5]. Он обеспечивает легкую настройку систем и выполнение. Также, программа структурировано сохраняет результаты и подготавливает `gnuplot`-скрипты для отрисовки графиков. Взаимодействие с программой происходит через аргументы командной строки и .json-файл с конфигурацией окружения модели Курамото и параметров, необходимых для моделирования системы. Данная программа поддерживает несколько настроек одновременно, что позволяет исследователю меньше контроллировать расчет. Для снижения времени разработки были использованы следующие библиотеки: «Qt»[7] и «JSON for Modern C++»[8].

[4]Исходный код библиотеки: https://bitbucket.org/kirillovoa/kpon.

[5]Исходный код библиотеки: https://bitbucket.org/kirillovoa/kponCLI

4 Численное моделирование

Расчеты производились с использованием технологии OpenCL на центральном процессоре ввиду большего количества ядер и большей производительности на ядро. Результаты усреднялись по 25-ти запускам.

4.1 Классическая сеть фазовых осцилляторов Курамото под внешним шумовым воздействием

При воздействии шума на классическую систему синхронный режим возникает при больших значениях параметра связи λ. Связанно это с тем, что шум привносит нелинейное поведение в процесс обмена энергиями двух отдельных осцилляторов. Это мешает синхронизации, и при большой интенсивности шума, синхронизация не наступает вовсе (рис. 4).

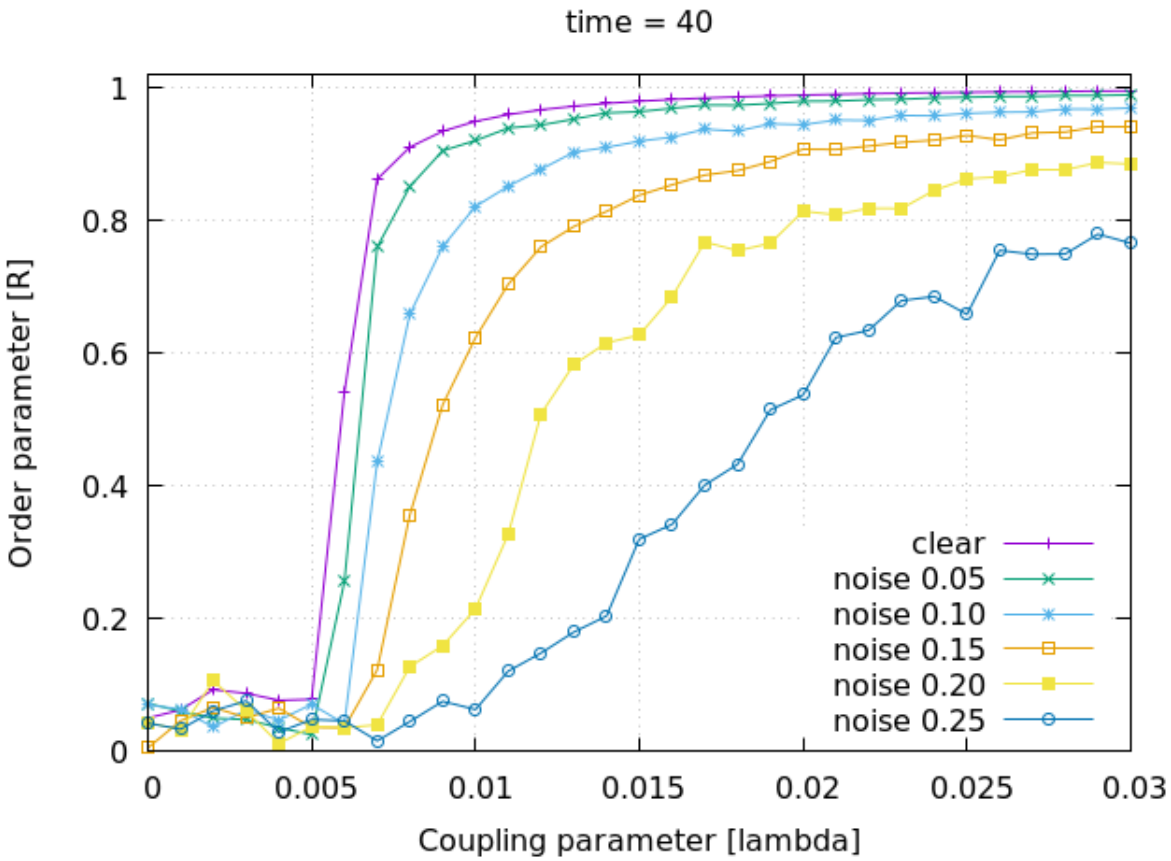

Рис. 4: Установление синхронизации в классической сети фазовых осцилляторов Курамото в присутствии шума. $time = 40$, $\lambda \in [0.00; 0.03]$, $D \in \{0.0, 0.05, 0.10, 0.15, 0.20, 0.25\}$

4.2 Адаптивная сеть фазовых осцилляторов Курамото под внешним шумовым воздействием

Как было сказано выше, адаптивная модель больше подвержена кластеризации и более устойчива к незначительным колебаниям фаз осцилляторов, но случайное возмущение, сильнее некоторого критического значения, инициирует смену состояния. При воздействии шума на систему, вероятность того, что следующее изменение фазы отдельного осциллятора будет больше критического значения увеличивается. В следствии этого, переход из несинхронного состояния в синхронное происходит при меньших значениях параметра связи λ для прямого направления, и больших для обратного направления, что можно наблюдать на рис. 6.

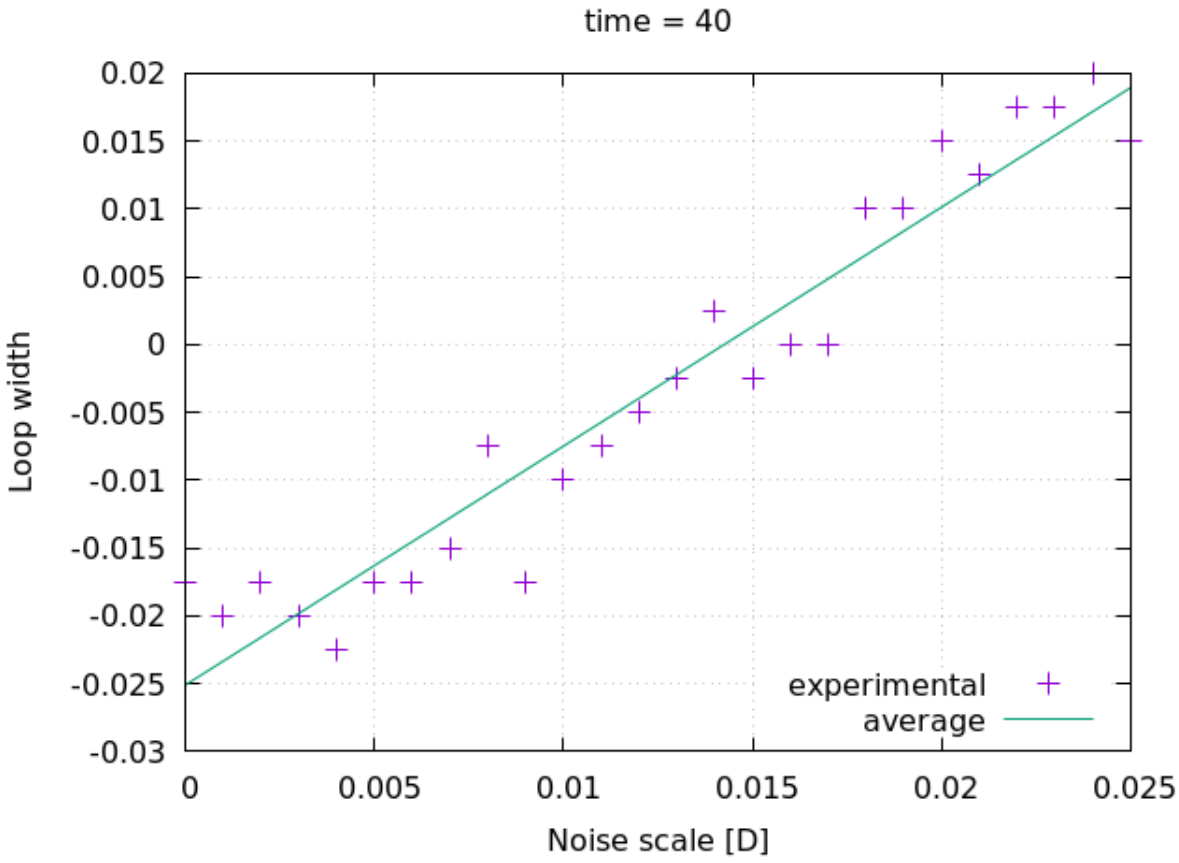

Рис. 5: Зависимость ширины петли от интенсивности шумового воздействия. $time = 40$, $D \in [0.000; 0.025]$

Как можно видеть на рис 6, критические значения параметра связи λ при которых происходит смена состояния сети различаются и на графиках появляется т.н. петля (рис. 5).

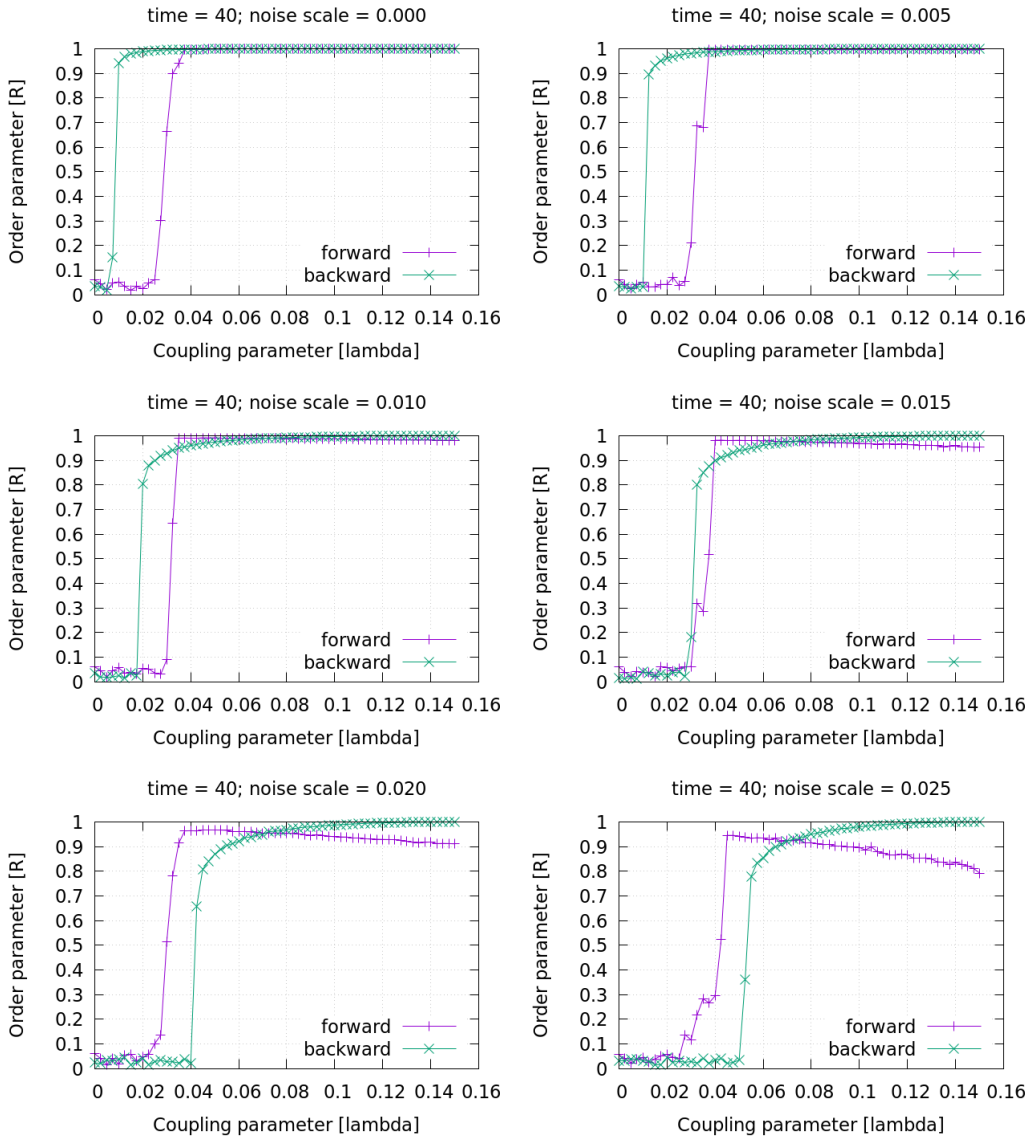

Рис. 6: Установление синхронизации в адаптивной сети фазовых осцилляторов Курамото в присутствии шума. $time = 40$, $\lambda \in [0.00; 0.15]$, $D \in \{0.000, 0.005, 0.010, 0.015, 0.020, 0.025\}$

Судя по результатам, можно утверждать, что изменение критического значения параметра связи, при котором происходит изменение состояния системы, линейно зависит от шумового воздействия на систему.

Заключение

В ходе бакалавской работы были подробно разобраны технологии параллельных вычислений OpenCL и OpenMP и базовые примеры к ним. Данные примеры можно использовать как для составления прикладных программ с использованием данных технологий, так и для методического пособия по углубленному изучению данных технологий.

Был разработан программный пакет для изучения синхронизации в сетях фазовых осцилляторов Курамото. Данную библиотеку можно использовать обособленно, через интерфейс командной строки, или в более крупном проекте. На данный момент доступны следующие модели систем: классическая, классическая с шумом, адаптивная, адаптивная с шумом. В дальнейшем планируется расширение круга моделей, реализация многослойных систем и дополнительных встроенных средств анализа систем.

Также было изучено поведение классической и адаптивной систем под воздействием шума. Показано, что шумовое воздействие на классическую систему увеличивает диапазон параметров связи λ, при которых наступает лишь частичная синхронизация. Для адаптивной модели была получена зависимость изменения ширины петли от интенсивности шума. Данная зависимость описывает общее поведение и является базисом для дальнейших исследований влияния шума на адаптивную модель сети фазовых осцилляторов Курамото.

Список используемых источников

[1] Explosive Synchronization in Adaptive and Multilayer Networks / Xiyun Zhang, Stefano Boccaletti, Shuguang Guan, and Zonghua Liu // Physical Review Letters — 2015. — № 114. — с. 038701-1

[2] OpenMP. Главная страница. [Электронный ресурс]. Режим доступа: https://www.openmp.org/. (Дата обращения: 22.05.2018).

[3] OpenMP 4.5. Complete Specifications. [Электронный ресурс]. Режим доступа: https://www.openmp.org/wp-content/uploads/openmp-4.5.pdf. (Дата обращения: 22.05.2018).

[4] Khronos Group. Главная страница проекта OpenCL. [Электронный ресурс]. Режим доступа: https://www.khronos.org/opencl/. (Дата обращения: 22.05.2018).

[5] OpenCL 1.2 specifications. Revision 19. [Электронный ресурс]. Режим доступа: https://www.khronos.org/registry/OpenCL/specs/opencl-1.2.pdf. (Дата обращения: 22.05.2018).

[6] OpenCL 2.2-7 specifications. [Электронный ресурс]. Режим доступа: https://www.khronos.org/registry/OpenCL/specs/2.2/pdf/OpenCL_API.pdf. (Дата обращения: 22.05.2018).

[7] Qt Framework. Главная страница. [Электронный ресурс]. Режим доступа: https://www.qt.io/. (Дата обращения: 22.05.2018).

[8] JSON for Modern C++. Репозиторий. [Электронный ресурс]. Режим доступа: https://github.com/nlohmann/json. (Дата обращения: 22.05.2018).