МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение высшего образования
«РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ ГИДРОМЕТЕОРОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Институт информационных систем и геотехнологий

Кафедра ФИЗИКИ

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Исследование многослойных наноструктур методом нейтронной рефлектометрии.

Направление подготовки 030302 «Физика»

(уровень бакалавриата)

Профиль «физика»

Студент Э.И.Дьячук

Научный руководитель доцент кафедры физики Хлябич П.П.

Консультант ВКР,

Старший научный сотрудник Национального Исследовательского Центра «Курчатовский институт»-Петербургского Института Ядерной Физики,

канд. физ.-мат. наук, доцент_В.Г. Сыромятников

Санкт-Петербург, 2022

Оглавление

Введение

Такой метод исследования как нейтронная рефлектометрия, зародился в процессе проведения многочисленных экспериментов, связанных с растущим интересом к слоистым тонкопленочным наноструктурам. Данное явление объяснимо: интерес подкреплен развитием нанотехнологий, позволяющих приходить к все более оптимальным и компакным решениям многих как технических, так и научных задач.

Нейтронная рефлектометрия находит себе применение не только в квантовых исследованиях и созданиях нового оборудования, но и в электронной и оптической промышленности.

В методе проведения эксперимента нейтронной рефлектометрии используются два типа нейтронных пучков: монохроматические и «белые» пучки, т.е. с широким спектральным распределением. В случае с монохроматическим пучком для сканирования обратного пространства используются измерения на различных углах скольжения. А для второго вида пучка используется всего один, фиксированный угол скольжения, для него измеряются сечения рассеяния в зависимости от нейтронной длины волны по время-пролетной методике. Однако, если следует отсканировать более широкую область измерения по время-пролетной методики могут проводиться так же и на нескольких углах скольжения. Поэтому использование монохроматических и «белых» нейтронных пучков, позволяет отвечать на одни и те же задачи исследования [2].

Целью данной выпускной квалификационной работы является изучение методологии исследования образцов наноструктур с использованием нейтронного пучка, имеющего широкое спектральное распределение методом нейтронной рефлектометрии.

Глава 1. Теоретические сведения.

1.1. Оптические свойства нейтрона.

Цель исследований – это измерить и представить в численном виде коэффициент отражения от среды нейтронов, не как частиц, а как волны, описываемой функцией переданного импульса.

Конечно, нейтрон, как отдельная частица имеет свои характеристики, играющие большую роль в расчетах. К таким относятся: масса, скорость, магнитный момент, переданный импульс. По корпускулярно-волновой теории, нейтрон так же и волна – волна де Бройля, поэтому все эти характеристики могут быть связаны простейшими формулами:

$$\lambda = \frac{h}{P}, \quad (1)$$

где P – импульс, h - постоянная Планка

или, раскрывая, $\lambda = \frac{h}{mv}$, (2)

где mv – импульс в виде произведения массы на скорость нейтрона.

Так как в дальнейшем мы будем рассматривать не стационарное состояние частиц, а их движение, то это стоит учесть с помощью кинетической энергии E:

$$E = \frac{mv^2}{2}, \quad (3)$$

Так как мы имеем дело с целым нейтронным пучком, следует учесть волновой вектор, определяющий направление переноса этой энергии:

$$k = \frac{2\pi}{\lambda}, \quad (4)$$

Теперь с учетом (4) уравнение (3) может принять вид:

$$E = \frac{h^2 m}{2\lambda^2 m^2} = \frac{h^2}{2\lambda^2 m} = \frac{k^2 h^2}{2m}, \quad (5)$$

Теперь для уточнения, что нейтронный направленный пучок взаимодействует со средой будет использоваться уравнение Шредингера в стационарном ввиде.

$$\frac{\hbar^2}{2m}\nabla^2\psi + (E - U_r)\psi = 0, \quad (6)$$

Где $\psi(x, y, z)$ – пси-функция, решение уравнения Шредингера, а

U_r – нейтронно-оптический потенциал.

Далее можно свести одномерную задачу к задаче с отражением частицы от потенциального барьера для этого рассмотрим рисунок:

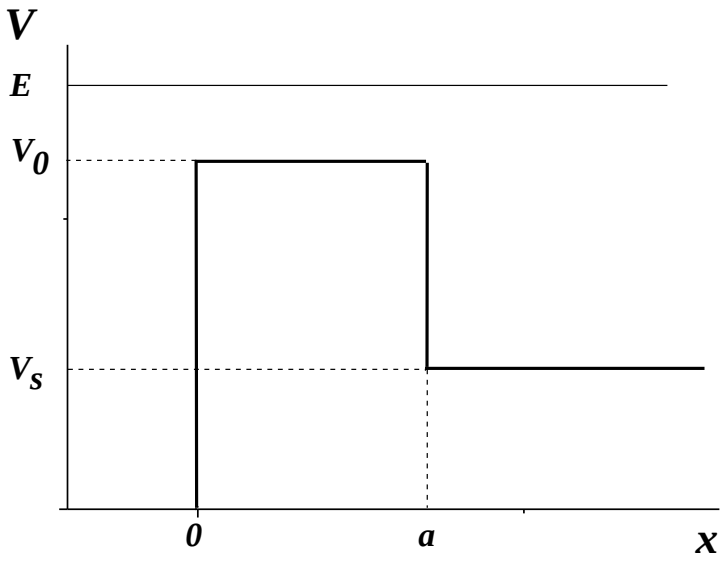


Рис.1. Зависимость нейтронно-оптического потенциала пленки от координаты z, направленной от поверхности пленки в глубину. V_0 и V_s - потенциалы пленки и подложки, соответственно; a - толщина пленки.

Далее рассмотрим рисунок отображающий эксперимент по отражению нейтронов от поверхности с помощью рефлектометра.

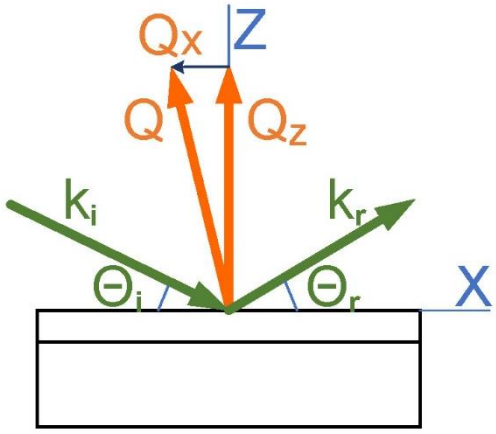


Рис.2. Схема эксперимента по нейтронной рефлектометрии.

Так как задача одномерная и мы выбрали направленную вертикально ось z, то компоненты для волнового вектора k, фигурирующего в формуле (5) и, соответственно (6) можно спроецировать на эту ось. Так нормальная компонента волнового вектора:

$$k_z = k \sin(\theta) = \frac{2\pi \sin(\theta)}{\lambda} = q \quad (7)$$

Он же будет являться половиной от значения изменения волнового вектора нейтрона или переданного импульса, изображенного на Рис.2 как Q.

Теперь уравнение Шредингера (6) может принять иной вид:

$$\frac{\hbar^2}{2m}\psi'' + \left(\frac{\hbar^2}{2m}q^2 - U_z\right)\psi = 0 \quad (8)$$

Собственные конечные значения данной функции будут решением уравнения, в частности при различных условиях отражения, например, в простейшем случае, от полубесконечной немагнитной среды.

В данном случае выведенное уравнение Шредингера (8) претерпевает изменения. Зададим граничные условия. В случае рассмотрения действия в отрицательной части оси, на которую спроецировано уравнение:

$\frac{\hbar^2}{2m}\psi"(z) + \frac{\hbar^2}{2m}q^2\psi(z) = 0$, (9)

И в случае рассмотрения в точке ноль или положительной части оси:

$\frac{\hbar^2}{2m}\psi" + \frac{\hbar^2}{2m}(q^2 - u)\psi = 0$, (10)

Этот случай означает преломление проходящего пучка.

По аналогии с оптикой можно ввести коэффициент преломления n:

$n = \frac{q\prime}{q} = \sqrt{1 - \frac{u}{E}} \cong 1 - \frac{pb_c}{2\pi}\lambda^2$, (11)

где ρ - концентрация ядер нанослоя (пленки) в 1 см^3, b_c - длина когерентного ядерного рассеяния

Можно ввести угол полного внешнего отражения путем преобразования формулы (12)

$\cos\theta_c = n$, (12)

$\theta_c = \lambda\sqrt{\frac{pb_c}{\pi}}$, (13)

соответствующий ему критический переданный импульс:

$Qc = \frac{4\pi sin\theta_c}{\lambda} = 4\sqrt{\pi pb}$, (14)

Так же можно из уравнения (8) получить амплитуду пропускания t и отражения r

$t = \frac{2q}{q+q\prime}$, (15)

$r = \frac{q-q\prime}{q+q\prime}$, (16)

где q’ – нормальная компонента отраженного пучка

В соответствии с полученным уравнением (16) коэффициент отражения нейтрона от нанослоя (пленки) равен:

$$R=|r|^2 = \left|\frac{q-\sqrt{q^2-u}}{q+\sqrt{q^2-u}}\right|^2, \quad (17)$$

Если критический импульс будет численно больше переданного, коэффициент отражения будет равен единице и будет наблюдаться полное отражение.

1.2. Наноструктуры.

Нейтроны обладают уникальными характеристиками, что отличает их использование в исследованиях от применения других частиц. Во-первых, энергия нейтронов, из-за наличия у них массы, значительно меньше, чем энергия рентгеновских и гамма-лучей при той же длине волны и эта энергия оказывается сравнимой с энергией тепловых колебаний атомов и молекул в веществе, что дает возможность изучать не только усредненную статическую атомную структуру вещества, но и динамические процессы, в нем происходящие.

Во-вторых, нейтрон обладает магнитным моментом, и это позволяет изучать магнитную структуру, магнитные возбуждения, что, как оказалось, весьма существенно для понимания природы и процессов, происходящих, например, в таких важных материалах, как высокотемпературные сверхпроводники.

В-третьих, нейтроны взаимодействуют с атомными ядрами, что обусловливает их большую "контрастность" (чувствительность) в различении атомов, близко расположенных в таблице Менделеева элементов. Особенно

это относится к легким элементам (водород, кислород и др.), идентификация которых в телах, содержащих тяжелые элементы, почти невозможна рентгеновскими и гамма-методами, а именно их положение часто определяет свойства материала. Нейтронам же доступно изучение изотопного состава вещества.

Кроме того, нейтроны электрически нейтральны, и их взаимодействие с ядрами является слабым, что позволяет им достаточно глубоко проникать в вещество – в этом их существенное преимущество, по сравнению с рентгеновскими и гамма-лучами, а также пучками других заряженных элементарных частиц.

Кроме того, не смотря на свою высокую проникающую способность, нейтроны не будут разрушать структуру вещества.

В связи с этим, методы нейтронных исследований позволяют развивать направления в изучении и создании различных наноструктур и наноматериалов [4]. К ним относятся:

— Нанокластеры металлов – соединения одного или двух атомов, образующих наноструктуру;

— Ультрадисперсные алмазы – искусственно выращенные, посредством воздействия высоких энергий от взрывов, алмазы

— Фуллерены – «нано-кирпичи» из котрых можно собрать новые структуры и материалы

— Ферримагнитные материалы – используемые в магнитозаписывающих устройствах

— Центров кристализации – наночастиц и др.

А также методы нейтронных исследований могут применяться в

— Инженерной диагностике

— Молекулярной биологии и фармакологии для решения проблем безопасности лекарствнных средств и разрешении задач внутри РНК и ДНК

1.3. Рефлектометр и результат его работы, как метод исследования наноструктур.

В данной работе предоставлен рефлектометр THP - это комбинированный рефлектометр, т.к. он работает, как по время-пролетному методу, так и с фиксированной длиной волны. Он создан полностью на базе рефлектометра HP-4M [5].

Преимущества этих методов объединены в данном нейтронном рефлектометре для оптимизации времени измерения для различных типов образцов. В каждом из методов мы можем использовать либо неполяризованный, либо поляризованный пучок. Таким образом, рефлектометр имеет четыре режима работы: “белый” неполяризованный пучок (режим I), “белый” поляризованный пучок (режим II), монохроматический неполяризованный пучок (режим III), монохроматический поляризованный пучок (режим IV). Более того, переход из одного режима в другой довольно прост и не требует какого-либо изменения геометрии измерений. Исследования на рефлектометре можно проводить путем измерения зеркального отражения и незеркального рассеяния как неполяризованных, так и поляризованных нейтронов. В частности, незеркальное рассеяние нейтронов позволяет определять параметры неидеальности многослойных образцов, такие как шероховатость. Схема рефлектометра THP изображена на Рис. 3.

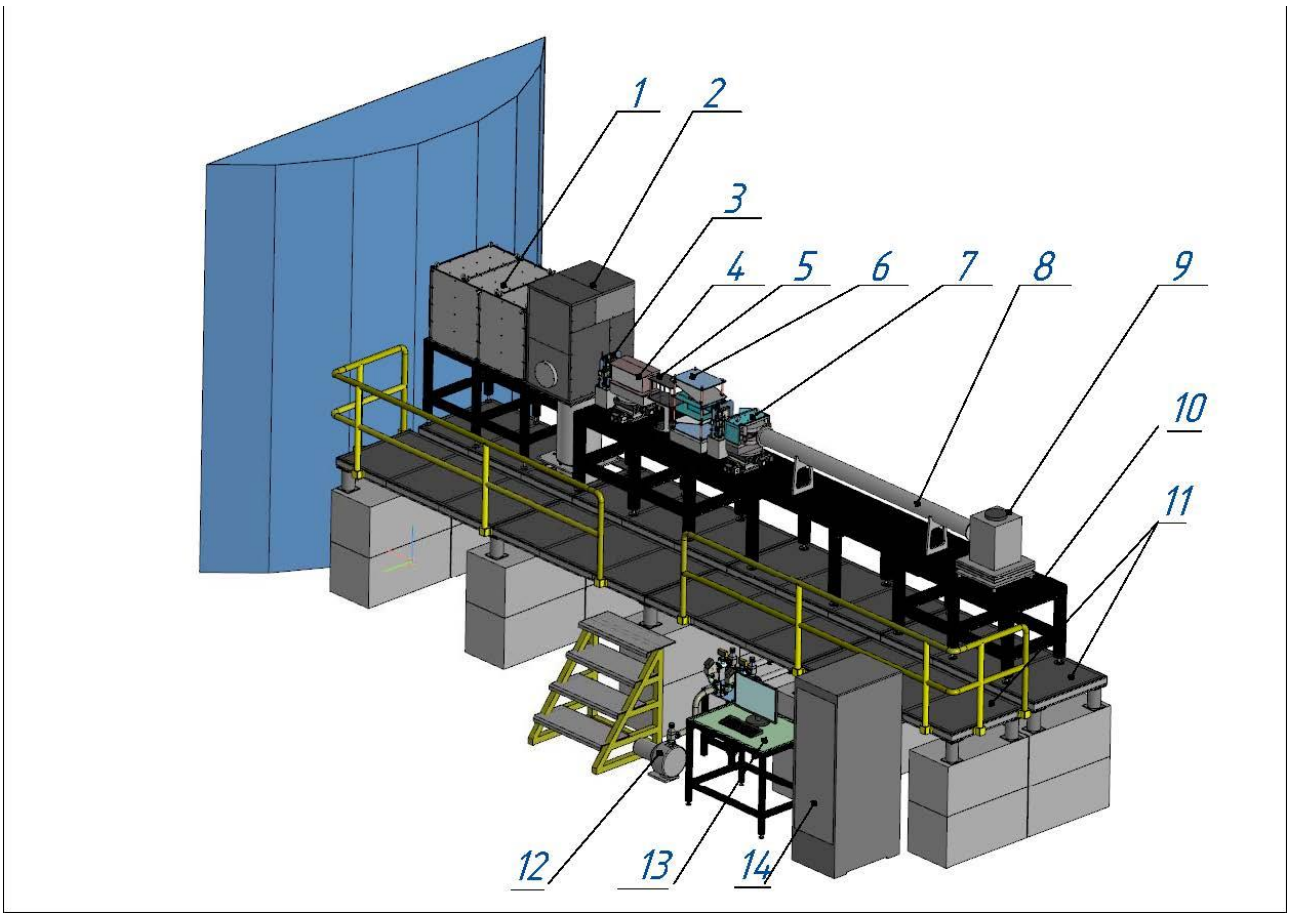


Рис. 3. 3D схема нейтронного рефлектометра ТНР. 1. Защита отклоняющего (фильтрующего) пучок зеркала с фоновым коллиматором; 2. Защита прерывателя нейтронного пучка с заслонкой пучка; 3. 2D диафрагма, задающая ширину и высоту пучка, падающего на вход нейтронно-оптического формирователя пучка; 4. Нейтронно-оптический формирователь пучка на юстировочном столе; 5. Магнитная система для создания ведущего магнитного поля; 6. Радиочастотный спин-флиппер; 7. Узел образца с электромагнитом; 8. Вакуумный тракт; 9. Блок детектора с коллимационной диафрагмой на юстировочном столе; 10. Базисная немагнитная приборная платформа; 11. Виброустойчивый приборный стапель и стапель для персонала; 12. Вакуумный насос; 13. Управляющий компьютер; 14. Шкаф с электроникой.

Глава 2. Моделирование и проведение эксперимента.

Для создания модели и проведения эксперимента были использованы перечни программ, описание которых представлено ниже.

2.1. Программная среда Microsoft Visio.

Данная среда является довольно интуитивным редактором для создания схем и диаграмм, чертежей, блок-схем с помощью встроенного блока конструктора, с включенной в себя библиотекой визуальных объектов также может служить в качестве уникального инструмента для представления графических контейнеров, выносок, соединительных линий, текстовых полей, ссылок и других объектов в наглядной, понятной визуализированной форме

2.2. Методы Монте-Карло.

При создании нейтроноводов и нейтронных станций особое внимание уделяется предпроектным расчетам и глубокой оптимизации. Это объясняется как высокой стоимостью создаваемых изделий, так и относительно низкой светосилой нейтронных источников, которая вынуждает бороться даже за небольшие выигрыши в нейтронном потоке на образце.

Способы моделирования нейтронных станций условно можно разбить на три группы: аналитические, диаграммные и на основе метода Монте-Карло.

При аналитическом подходе интенсивность нейтронного пучка на выходе установки представляется произведением начальной интенсивности и функций пропускания каждого из элементов установки. При наличии большого числа оптических элементов в приборе результатом такого рассмотрения являются очень громоздкие выражения для интенсивности и

разрешения, зачастую плохо поддающиеся анализу. Альтернативным подходом к разработке нейтронных инструментов является комбинация методов трассировки пучков и Монте-Карло, чье широкое распространение стало возможным с развитием вычислительной техники. Из фазового объема, ограниченного соответствующими параметрами источника, случайным образом выбираются начальные траектории нейтронов. Далее они трассируются через всю оптическую систему, при необходимости их свойства вновь изменяются случайным образом (например, при дифракции на мозаичном кристалле). При условии достаточно большого количества траекторий и равномерной их выборки на выходе будет получено достоверное и точное описание нейтронного пучка.

Созданная таким методом модель может использоваться для решения различных задач:

- Получение достоверной оценки нейтронного потока на выходе оптической системы;
- Анализ распределения нейтронов по сечению и расходимости пучка, их спектра и поляризации;
- Оптимизация параметров установки по заданному критерию (обычно максимальная светосила при некоторых условиях);
- Проведение виртуальных экспериментов, облегчающих реальную работу экспериментаторов или использующихся в образовательных целях;
- Выяснение происхождения необъясненных эффектов при работе установки.

Использование метода Монте-Карло позволяет относительно легко моделировать сложные с точки зрения геометрии и содержащие большое число оптических элементов системы. Результаты симуляций представляются в наглядной и удобной для анализа форме.

2.3. Программная среда Mcstas.

Исторически, программные пакеты, реализующие метод Монте-Карло для нужд нейтронного рассеяния, создавались независимо во многих институтах,
зачастую под конкретную задачу.

Расчеты в данной работе выполнены с помощью пакета McStas, к чьим достоинствам также относятся отличная техническая поддержка, открытость исходного кода и удобный графический интерфейс.

Центральным элементом McStas является высокоуровневый язык программирования на основе языка C, использующийся для описания нейтронных установок. При запуске вычислений файл-описание транслируется на язык C и компилируется в исполняемый файл. Результаты его выполнения содержат информацию о нейтронном потоке в различных частях прибора, которые могут быть представлены в большом числе форматов, в том числе в виде текстовых файлов. Пакет включает в себя также графический пользовательский интерфейс, встроенное средство визуализации инструмента и результатов и несколько вспомогательных программ. Имеется поддержка параллельных вычислений с использованием технологий SSH (Secure Shell, безопасная оболочка) и MPI (Message Passing Interface, интерфейс передачи сообщений). Любая оптическая система в McStas моделируется как последовательность компонентов. Важной их особенностью является сочетание геометрического и физического описания моделируемого объекта в одном коде, что налагает некоторые ограничения на моделируемые системы. Все компоненты можно разде лить на три группы.

Компонент типа «источник» служит для генерации нейтронных траекторий. Каждой из них случайным образом сопоставляется набор величин — начальные координаты r, скорость v, поляризация P, время рождения t и вероятностный вес p. Параметры r и v ограничены геометрией источника и параметрами замедлителя. Время рождения t используется для установок,

использующих время-пролетную методику. У каждой только что созданной траектории p = 1.

Большинство последующих компонентов, которые моделируют зеркала, нейтроноводы, монохроматоры, прерыватели и пр., принимают параметры траекторий в качестве входных, меняют их согласно заданным законам рассеяния и передают следующую компоненту. Здесь демонстрирует высокую эффективность применение вероятностного веса p. Например, если нейтроны сталкиваются с зеркалом, вероятность отражения от которого составляет 50%, то в соответствии со строгим рассмотрением половина нейтронов должна быть поглощена, а половина — отражена.

2.4. Библиотека iFit

Важной особенностью McStas является интеграция с MATLAB посредством
библиотеки iFit.

Во-первых, такая интеграция помогает быстро обрабатывать полученные данные и визуализировать результаты.

Во-вторых, что более существенно, открывается возможность к оптимизации нейтронных инструментов по заданному критерию в многопараметрических пространствах. Библиотека iFit содержит реализации большого количества оптимизационных алгоритмов. Так как результаты расчетов методом Монте-Карло являются сравнительно «шумными», далеко не все широко распространенные методы многопараметрической оптимизации оказываются подходящими.

2.5. Программная среда Matlab

Программная среда Matlab является собирательным пакетом математических функций и операторов, направленных на сложные вычисления. Создание проектов возможно на встроенном с потенциалом к интерпретации языке MATLAB, выстроенном на матрицах структур данных, оперирующих различными спектрами объекто-ориентированных функций. Данная среда разработки позволяет написать код и выстроить зависимые графики в различных ориентациях плоскости или объемов, а так же, что немаловажно имеет возможность связи с другими программами. В данной работе, графики простейшей модели были выведены с помощью возможности Matlab.

2.6. Программная среда UNISON.

Для реализации модульной структуры программного обеспечения для автоматизации эксперимента, была составлена компьютерная программа UNISON [8] в среде Windows для управления блоками рефлектометра, сбора и обработки данных. Модульная структура программного обеспечения предоставляет возможность работы со сравнительно независимыми частями программы, каждая из которых вызывается командой. Таким образом, каждая команда соответствует одной или нескольким операциям с алгоритмической и, при необходимости, электронной поддержкой.

Каждая команда имеет имя и свои параметры - определяющие аргументы. Выполнение команды соответствует выполнению логически завершенной операции. Чтобы команды были совместимы друг с другом, они сопоставляются по параметрам. Программа обеспечивает ввод команд, их издание и выполнение в заданном порядке и т.д., и включает в себя основные команды общего характера и специальные команды, разработанные для определенного инструмента и его аппаратного обеспечения. Следует отметить, что алгоритм специальной команды может быть очень простым

(скажем, команда FLIPPER_1 включает/выключает flipper 1) или довольно сложным (например, команда ADJUST_MIRROR выполняет автоматическую настройку образца зеркала). Команды могут выполняться либо сразу после ввода, либо в группе с другими командами. В последнем случае команды либо вводятся с клавиатуры, либо считываются из файла, а затем выполняются в соответствии с правилами, определенными в ядре программы.

Серия таких команд формирует последовательность команд, которая может быть выполнена, просмотрена, отредактирована, отменена полностью или частично, записана в файл (последовательности команд, записанные в файл, как правило, соответствуют типичным измерениям и рутинным процедурам). Выполнение последовательности команд может быть начато, прервано, продолжено или отменено. В любой момент можно получить экспресс-информацию о состоянии установки, о текущих значениях параметров и о ходе эксперимента, или вмешаться в выполнение последовательности команд, изменить параметры и т.д.

Алгоритмическая гибкость достигается отсутствием каких-либо ограничений на количество команд с независимыми аргументами, произвольностью последовательности команд, некоторыми другими возможностями, предоставляемыми базовыми командами и ядром программы. Почти каждая специальная команда представляет собой отдельное приложение со своим собственным окном, системным меню и меню пользователя и может выполняться за пределами программы UNICOM (UNICOM оперирует только названиями приложений). Как следствие, компиляция новых и модификация существующих специальных команд не требуют изменений в программе UNICOM. Необходимость включения новой команды или изменения существующей команды может возникнуть при обновлении оборудования или добавлении нового оборудования.

2.7. Программная среда Lab Windows.

LabWindows™/CVI — это интегрированная среда разработки ПО, использующая стандарт ANSI C и предоставляющая обширный набор инструментов программирования для решения задач тестирования и измерения.

Интегрированная среда программирования LabWindows™/CVI ANSI C помогает создавать специализированные инженерные приложения. Это позволяет использовать среду для управления проектом, редактирования и отладки исходного кода, создания интерфейса пользователя, вывода кода тестирования и повышения его производительности в одном оптимизированном рабочем пространстве с вкладками. LabWindows/CVI включает средства для расширенной отладки, документирования кода и развертывания системы, что позволяет интегрировать управление исходными кодами, требования и системы управления данными. Программное обеспечение также упрощает быстрый сбор данных с приборов GPIB, USB, последовательного интерфейса, Ethernet, PXI, VXI и приборов с ПЛИС с использованием встроенных библиотек ввода-вывода приборов, встроенных драйверов приборов или двух интерактивных помощников по проведению измерений. Торговый знак LabWindows используется по лицензии корпорации Майкрософт. Windows является зарегистрированным товарным знаком корпорации Майкрософт в США и других странах [6].

Моя задача заключалась в том, чтобы с применением программной среды Lab Windows/ CVI , написать программу в качестве дополнения к программе UNISON, способную считать полученные результаты работы нейтронного рефлектометра из файла с любым текстовым расширением, а так же вывести все полученные значение на экран с возможностью

построения наглядного графика, а так же настройки интерфейса этой программы.

Глава 3. Анализ и вывод расчетных данных.

3.1. Схематическое представление рефлектометра

Для проведения эксперимента следует эмулировать его на рефлектометре с помощью компьютера, что можно воплотить, построив собственную модель. Для начала обратимся к самой простейшей модели.

Обратимся к программе Microsoft Visio [7] для составления схемы конструкции.

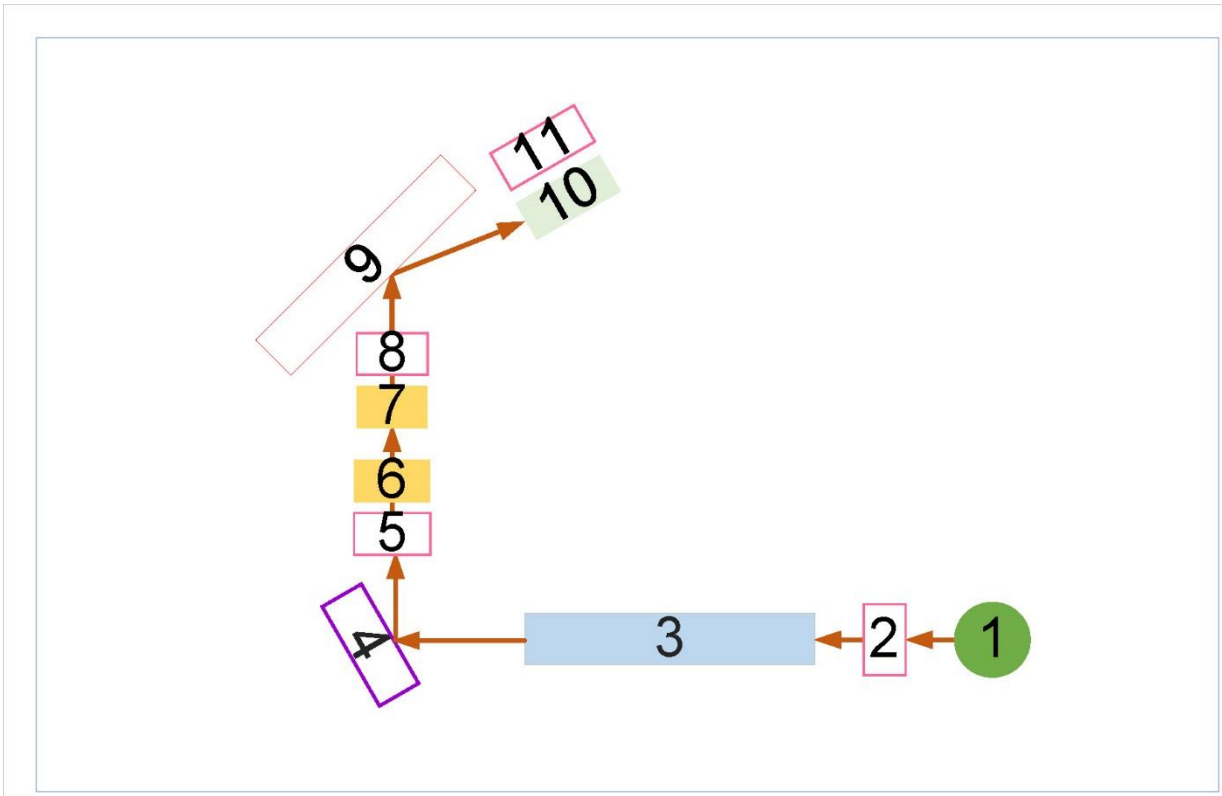


Рис. 4. Схематичная модель простого рефлектометра, составленная с помощью программы Microsoft Visio.

На схеме:

1. – Источник нейтронов
2. – Монитор № 1
3. – Нейтроновод
4. – Монохроматор
5. – Монитор № 2
6. – Диафрагма

7. – Диафрагма
8. – Монитор № 3
9. – Образец – суперзеркало Ni/Ti
10. – Детектор
11. – Монитор nD

Теперь, с помощью наглядного представления, задуманного, очевидно, по какому принципу будет идти нейтронный луч и чем будет зафиксирован. Теперь можно перейти к созданию и указанию точных параметров включений установки и проведения эксперимента с помощью программы Mcstas.

3.2. Создание программного кода для построения модели простого рефлектометра с помощью программы Mcstas.

Для проведения эксперимента с исследованием структуры с помощью методов рефлектометрии использовалась модель рефлектометра, написанная в программной среде Mcstas.

Код рефлектометра, использующего в данной работе описан в Приложении , ниже представлены его основные функции и описание входящих компонентов.

— DECLARE – объявление компонента, новой функции

— TRACE – вписываются входные данные состояния нейтрона по нескольким переменным, присваиваются значения переменны

— COMPONENT – вводятся новые компоненты с помощью данной задачи

— L_monitor() Этот компонент представляет собой одномерный монитор прямоугольной формы, который регистрирует истинную длину волны нейтрона и гистограммы входящей интенсивности на нескольких диапазонах длин волн.

— PSD_monitor() - создает двумерное изображение интенсивности нейтронов в реальном времени. Он часто используется в качестве детектора для приборов нейтронной визуализации.

— Arm() - наиболее часто используемый компонент McStas. Компонент должен использоваться в качестве ориентира для внутренней системы координат в McStas. Как правило, Рычаг() размещается на оси вращения, и следующие компоненты, включая дополнительные рычаги(), должны быть размещены относительно первого рычага(). Используемый таким образом компонент Arm() выполняет ту же функцию, что и оптический стенд.

— Guide() - Это нейтроновод с прямоугольными отверстиями. Вход имеет размеры h1 × w1, в то время как выход имеет размеры h2 × w2, т.е. с помощью него можно описать линейно сужающуюся направляющую, сделав прямоугольники входа и выхода разного размера. Длина направляющей задается буквой l.

— Monochromator_flat() - моделирует простой кристаллический монохроматор с размерами yheight × zwidth. Координата z здесь используется для обозначения ширины, поскольку невращающийся монохроматор лежит в плоскости (y, z). Это гарантирует, что угол поворота монохроматора станет равным углу θ по закону Брэгга

— Source_gen - представляет собой непрерывный источник нейтронов (прямоугольный или круглый). Угловое расхождение определяется размерами цели на которую подается поток. Форма может быть прямоугольной (размеры h – высота и w - длина) или дискообразная радиуса r.

— Slit – данный компонент представляет собой щель. При его объявлении создается отверстие при z = 0 и поток нейтронов будет направляться на эту плоскость. Действие данного объекта не затрагивать попавшие внутрь нейтроны, а остальные отбрасывать, впоследствии ими пренебрегая (они либо поглощаются материалом внешней каймы щели либо попросту ею отбрасываются)

— Monitor_nD – это общий монитор, который может выводить любой набор физических параметров, касающихся проходящих нейтронов. Сгенерированные файлы представляют собой либо набор одномерных. Входными параметрами для Monitor nD являются его размеры xmin, xmax, ymin, ymax (в метрах) и строка параметров, описывающая, что обнаруживать и что делать с сигналами.

— Mirror – это одиночная прямоугольная нейтронная зеркальная пластина. Ее можно использовать в качестве компонента образца или, например, для сборки полного нейтронного проводника путем размещения нескольких зеркальных компонентов в соответствующих местах и определенных ориентациях в пространстве. В локальной системе координат зеркало лежит в первом квадранте плоскости x-y, с одним углом в (0, 0, 0). Входными параметрами этого компонента являются размеры прямоугольного зеркала (l -ширина , h- высота) и значения R0, m, Qc, W и α для отражательной способности зеркала. В качестве частного случая, если m = 0, то отражательная способность равна нулю для всех Q, т.е. поверхность полностью поглощает входящий в нее поток. В этом случае нельзя анализировать прошедший через образец пучок.

— ReflectedBeam – отраженный пучок

— TransmittedBeam - поглощённый пучок или как вернее прошедший через образец пучек.

Далее приведены параметры перечня компонентов для смоделированной установки:

- источник нейтронов, модель реакторного канала для вывода пучка
- 10 метров нейтроновода 5x100 мм2, на боковых стенках m=1, на верхней и нижней m=2, по которому пущен пучок нейтронов.
- плоский монохроматор 300x120 мм2, PG, мозаичность 20' в обеих плоскостях, вращение вокруг вертикальной оси, последующие узлы автоматически вращаются на удвоенный угол, позволяющий выделять монохроматическое излучение из спектра
- пара щелей 5x100 мм2 на расстоянии 2м

- образец - зеркало m=3 2000x120 мм2, позиционировано по середине пучка, угол поворота отсчитывается от направления прямого пучка, вращение вокруг вертикальной оси, последующие узлы автоматически вращаются на удвоенный угол.
- монитор отражённого пучка (горизонтальный профиль)

Табл.1 Параметры для модели

Имя	**Ед. изм**	**Описание**	**Значения**
directbeam		Направления пучка	1 и 0
Lam_min	AA	Минимальная длина волны	4.95
Lam_max	AA	Максимальная длина волны	5.05
theta_m	градус	Скользящий угол	1

После запуска программы в назначенную эксперименту папку сохранились данные с аппаратуры, снимающей показания. Свяжем эти данные с помощью библиотеки iFit.

Для качественного отражения графика укажем путь построения через программную среду Matlab.

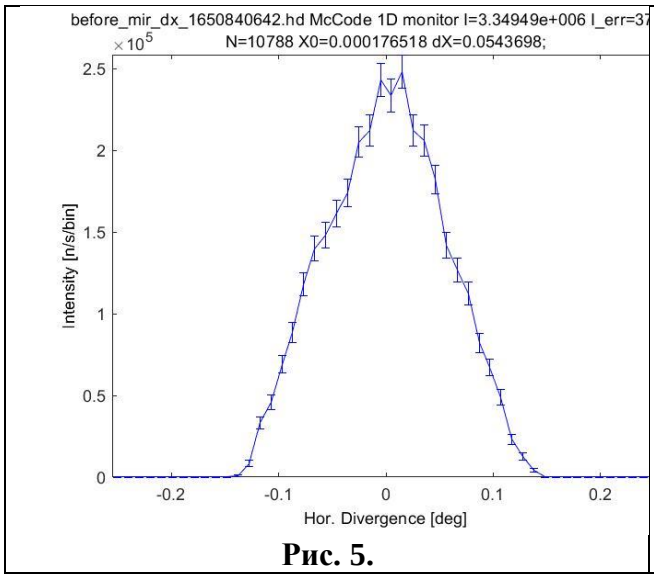


Рис. 5.

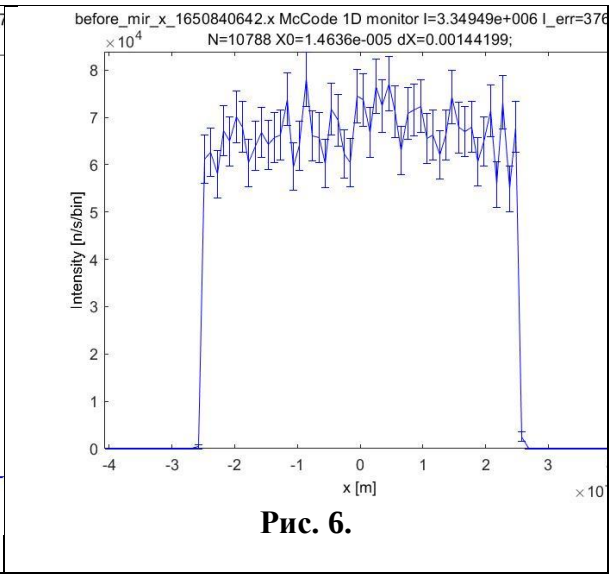


Рис. 6.

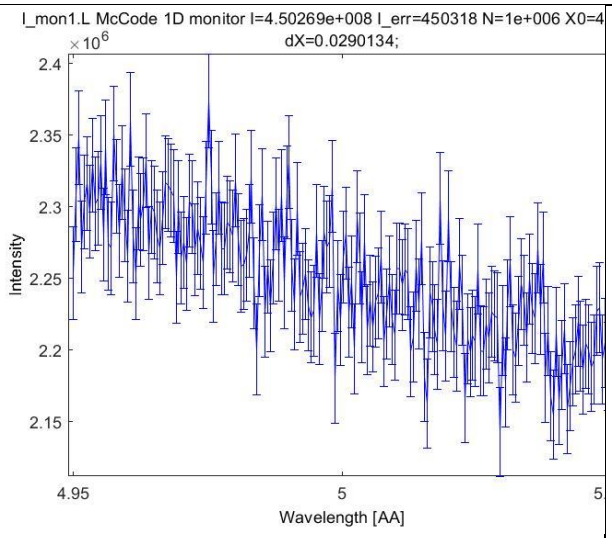


Рис. 7.

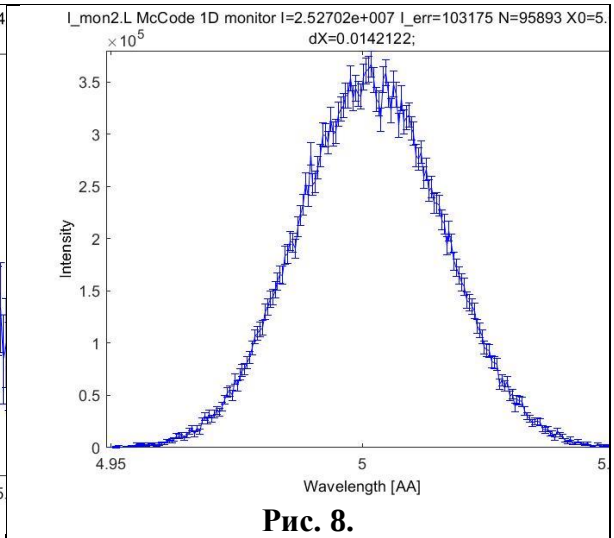


Рис. 8.

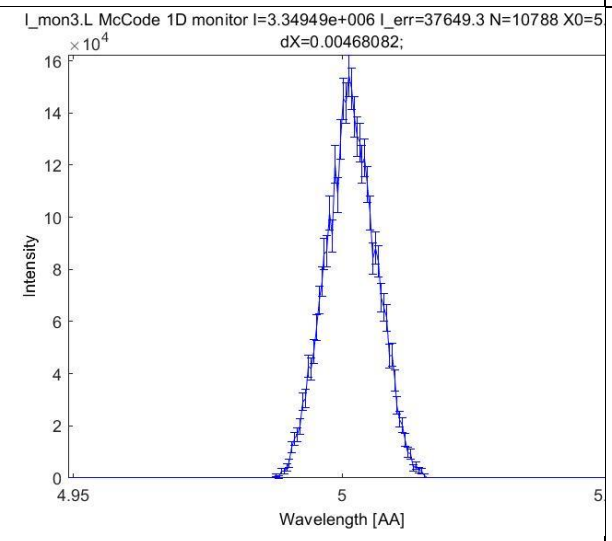


Рис. 9.

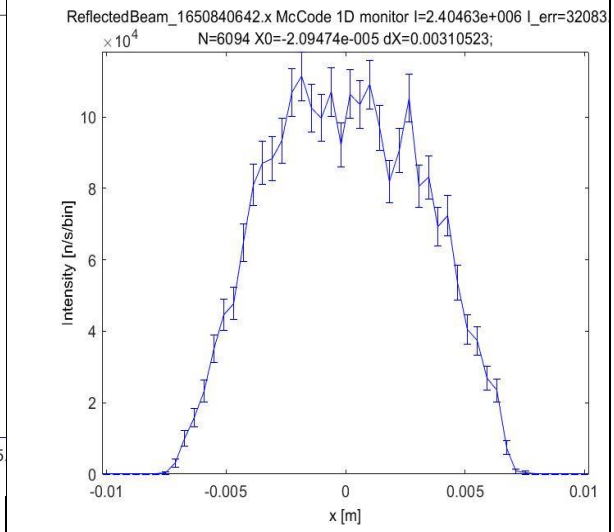


Рис. 10.

Рис.5. Угловое распределение интенсивности нейтронного пучка, падающего на образец.

Рис.6. Профиль пучка в горизонтальной плоскости на выходе из нейтроновода.

Рис.7. Зависимость интенсивности от длины волны, полученная на мониторе №1

Рис.8. Зависимость интенсивности от длины волны, полученная на мониторе №2.

Рис.9. Зависимость интенсивности от длины волны, полученная на мониторе №3.

Рис.10. Пространственное распределение интенсивности пучка, отраженного от образца суперзеркала Ni/Ti под углом 1 градус.

3.2.1. Установка нейтронного время-пролетного рефлектометра.

Теперь понимая структуру и механизмы работы простейшего рефлектометра, можно перейти к более сложной и рельной установке времяпролетного рефлектометра. Ее схематическое представление выполнено с помощью Microsoft Visio.

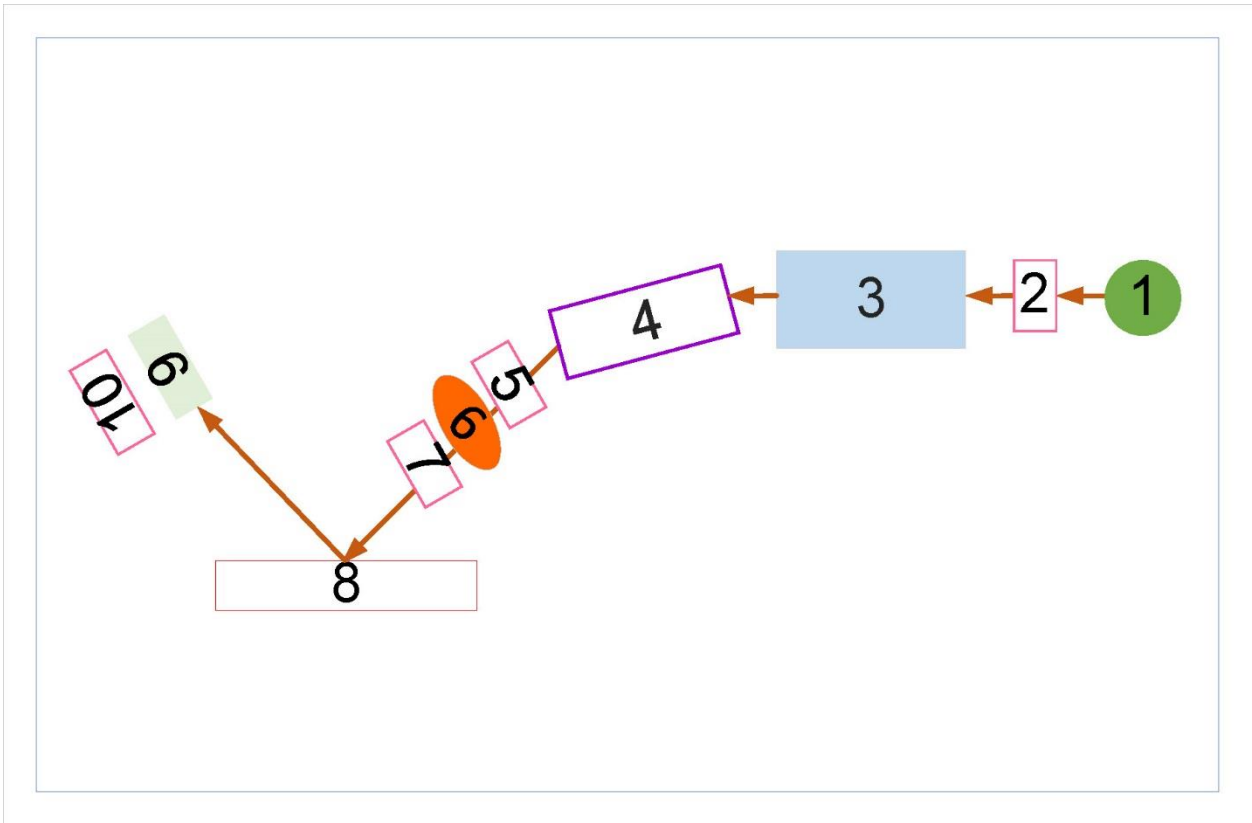


Рис. 11. Схема времяпролетного рефлектометра сделанная в Microsoft Visio.

На рис.11:

1. - Источник нейтронов
2. - Монитор № 1
3. - Коллиматор
4. - Отражающее суперзеркало
5. - Монитор № 2

6. - Прерыватель пучка (чоппер)
7. - Монитор № 3
8. - Образец
9. - Детектор
10. - Монитор nD

Для отображения многоканальных показаний обратимся к интегрированной среде разработки ПО CVI Windows и напишем программу с удобным пользовательским интерфейсом для графического вывода данных.

3.2.2. Структура программы Spectrum view

Моя программа называется «Spectrum view», это название выбрано не случайно. Spectrum view – в переводе с английского «Вид спектра», что говорит о прямом предназначении программы: анализ результата измерений зеркального отражения от образца является набор интенсивностей в зависимости от длины волны для данного угла скольжения (время-пролетный метод) или от угла скольжения для данной длины волны (метод измерений с фиксированной длиной волны).

На данном изображении указан полный вид программы для пользователя, интерфейс. А красными цифрами пронумерованы ее компоненты.

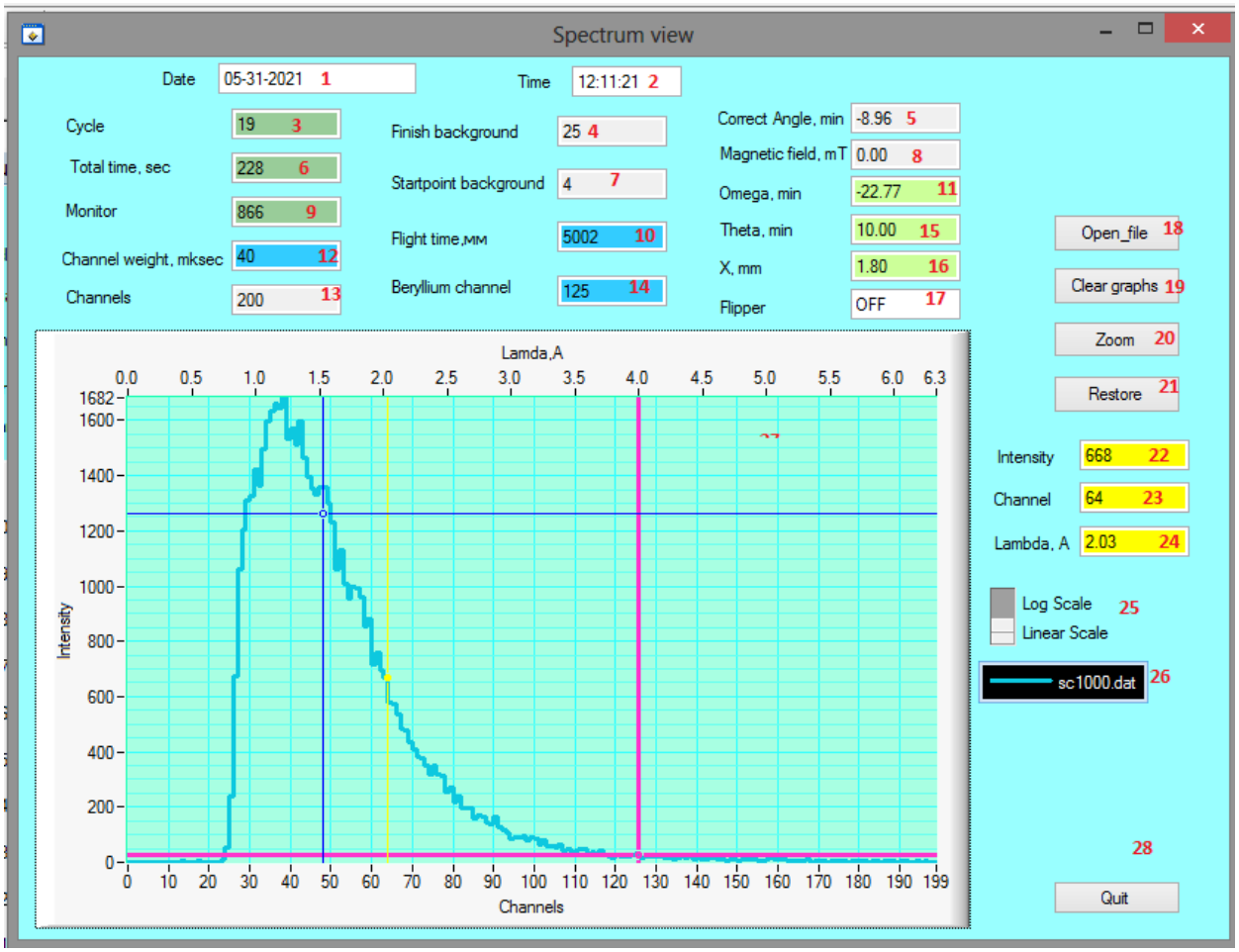


Рис. 12. Главное окно программы

1 - дата, 2 – время окончания измерения, 3 – общее число каналов, 4 – конечный канал диапазона измерения фона, 7 – начальный канал диапазона измерения фона, 9 – число мониторных импульсов за время измерения, 10 – время-пролетная база, 11 – координата детектора (омега) в угл. минутах, 14 – номер канала бериллиевого среза, 15 – угол скольжения образца (тэта) в угл. минутах, 16 – координата образца в направлении перпендикулярном оси пучка (X) в мм, 17 – состояние спин-флиппера (ON/OFF – включено/выключено), 18 – кнопка для открытия файла, 19 – кнопка очистки рабочей поверхности от графиков, 20 – кнопка приближения, зуммирования масштаба, 21 – кнопка сброса приближения масштаба , 22-24 – панель вывода значений, полученных курсором, привязанным к точке, 25 – изменение структуры графиков (линейный/ логарифмический), 26 – название открытого файла данных 27 – графическое представление полученных результатов с тремя курсорами :

синий и розовый – свободные, желтый – привязанный к точке, 28 – рабочая кнопка выхода их программы

3.2.3. Компоненты и оформление

Программа содержит три необходимых и изменяемых компонента.

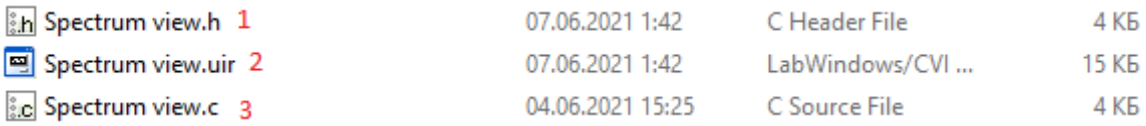


Рис. 13. Список файлов в проекте

1 – заголовочный h-файл, содержащий перечисление всех использованных интерактивных панелей и заголовки их callback функций (См. Приложение 2. Spectrum view.h) , 2 – uir-файл, содержащий в себе графические объекты программы (См. Приложение 3. Spectrum view.uir) , 3 – c-файл, содержащий в себе код программы (См. Приложение 4. Spectrum view.c) .

```
    /* Panels and Controls: */

#define  PANEL                          1
#define  PANEL_GRAPH                    2       /* control type: graph, callback function: readcoursor */
#define  PANEL_COMMANDBUTTON            3       /* control type: command, callback function: Quit */
#define  PANEL_NUMERIC_2                4       /* control type: numeric, callback function: (none) */
#define  PANEL_NUMERIC                  5       /* control type: numeric, callback function: (none) */
#define  PANEL_NUMERIC_3                6       /* control type: numeric, callback function: (none) */
#define  PANEL_NUMERIC_4                7       /* control type: numeric, callback function: (none) */
#define  PANEL_NUMERIC_5                8       /* control type: numeric, callback function: (none) */
#define  PANEL_NUMERIC_6                9       /* control type: numeric, callback function: (none) */
#define  PANEL_NUMERIC_7                10      /* control type: numeric, callback function: (none) */
#define  PANEL_NUMERIC_8                11      /* control type: numeric, callback function: (none) */
#define  PANEL_NUMERIC_9                12      /* control type: numeric, callback function: (none) */
#define  PANEL_NUMERIC_10               13      /* control type: numeric, callback function: (none) */
#define  PANEL_NUMERIC_11               14      /* control type: numeric, callback function: (none) */
#define  PANEL_NUMERIC_13               15      /* control type: numeric, callback function: (none) */
#define  PANEL_NUMERIC_14               16      /* control type: numeric, callback function: (none) */
#define  PANEL_NUMERIC_15               17      /* control type: numeric, callback function: (none) */
#define  PANEL_STRING                   18      /* control type: string, callback function: (none) */
#define  PANEL_STRING_2                 19      /* control type: string, callback function: (none) */
#define  PANEL_STRING_3                 20      /* control type: string, callback function: (none) */
#define  PANEL_COMMANDBUTTON_2          21      /* control type: command, callback function: openfile */
#define  PANEL_BINARYSWITCH_2           22      /* control type: binary, callback function: scale */
#define  PANEL_COMMANDBUTTON_3          23      /* control type: command, callback function: clear */
#define  PANEL_COMMANDBUTTON_4          24      /* control type: command, callback function: ZoomIn */
#define  PANEL_COMMANDBUTTON_5          25      /* control type: command, callback function: RestoreGraph */
#define  PANEL_NUMERIC_12               26      /* control type: numeric, callback function: (none) */
#define  PANEL_NUMERIC_18               27      /* control type: numeric, callback function: (none) */
#define  PANEL_NUMERIC_17               28      /* control type: numeric, callback function: (none) */


    /* Control Arrays: */

#define  CTRLARRAY                      1

    /* Menu Bars, Menus, and Menu Items: */

        /* (no menu bars in the resource file) */


    /* Callback Prototypes: */

int  CVICALLBACK clear(int panel, int control, int event, void *callbackData, int eventData1, int eventData2);
```

Рис. 14. Внешний вид файла с расширением «.h»

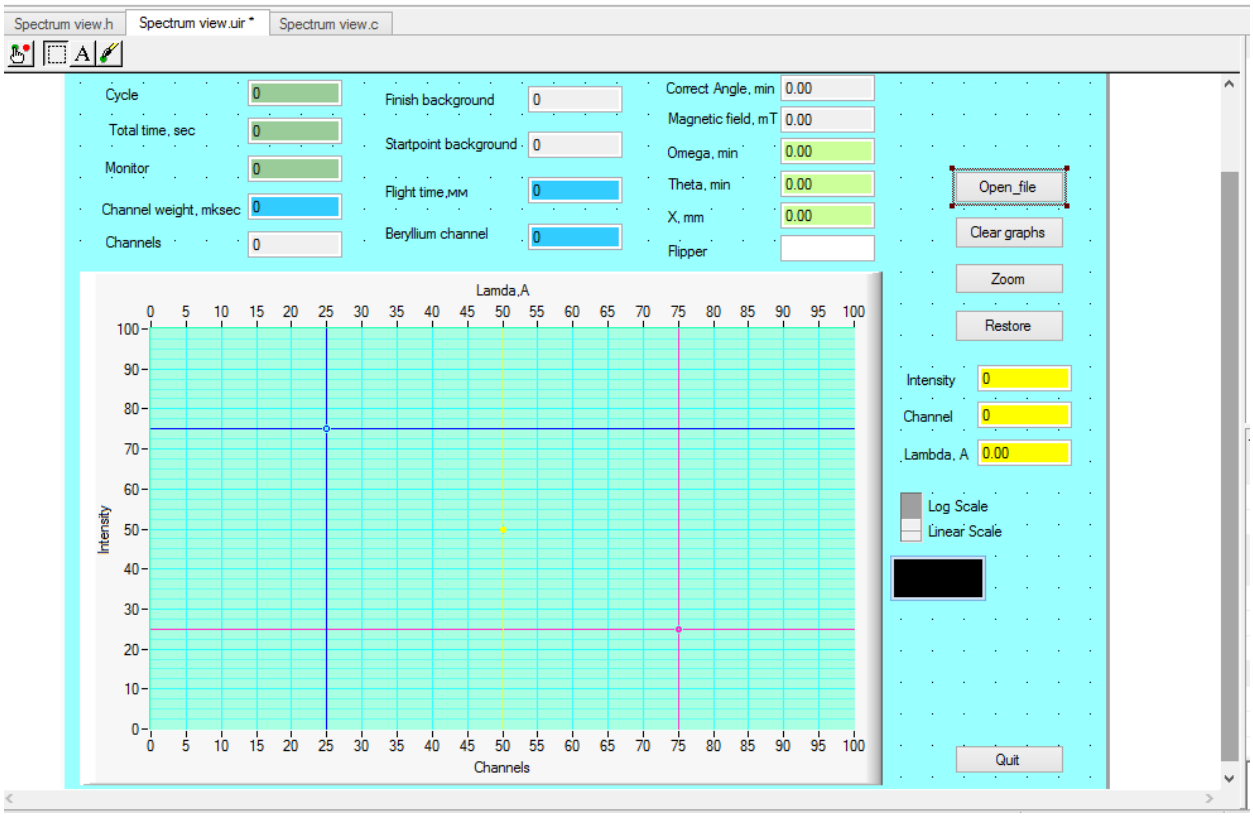


Рис. 15. Внешний вид файла с расширением «.uir»

Рис.16. Внешний вид файла с расширением «.c»

3.2.4. Листинг

Необходимый для непосредственного рассмотрения файл – «Spectrum view.c» , где содержится сам код программы.

```
#include <ansi_c.h>
#include <utility.h>
#include <cvirte.h>
#include <userint.h>
#include "Spectrum view.h"
```

Рис.17. Строчки, содержащие формулировку «#include» - обозначает подключение библиотек, содержащих набор необходимых компонентов, способных произвести успешный запуск программы.

Библиотека «Ansi» - является описанием реализации общих операций, таких как обработка ввода-вывода и строк, в языке программирования Си.

Библиотека «Utility» - основная библиотека функций-утилит

Библиотека «Cvirte» - обеспечивает работу приложений, созданных в LabWindows

Библиотека «Userint» - функции работы со всеми элементами интерфейса

Библиотека «spectrum view. H» - заголовочный файл, содержащий перечисление всех использованных интерактивных панелей и заголовки callback функций

```
static int panelHandle;
char f[255], n[255], fn[MAX_PATHNAME_LEN];
int channels;
int startbackground;
int  finishbackground;
float angle;
int cycle;
int summtime;
int monitor;
int Bechannel;
int flighttime;
int wightchannel;
float magneticfield;
char flipper[4];
float omega;
float theta;
float x;

char date[11];
char endtime[9];
int*graph;
int i;
int waveleight=0;
int plot=0;
int x1;
int y1;
int x2;
int y2;


 int read_file(char fn[]) ;
 int outgraph(int waveleight);
 void ReadIni (void);
 int ColorPlot [5]={0,0,0,VAL_RED,VAL_DK_CYAN};
```

Рис. 18. Перечень глобальных объявленных переменных.

Наименование, перед объявленными переменными – тип данных, категоризация аргументов операций над значениями, как правило, охватывающая как поведение, так и представление

Переменные, содержащие наименование static int – статический целый тип данных

Переменные, содержащие наименование int – простейший тип данных, для представления целых чисел

Переменные, содержащие наименование float – тип данных, отвечающих за значение с плавающей запятой

Переменные, содержащие наименование char – тип переменных, содержащих символьное значение

```
int main (int argc, char *argv[])
{
    if (InitCVIRTE (0, argv, 0) == 0)
        return -1;  /* out of memory */
    if ((panelHandle = LoadPanel (0, "Spectrum view.uir", PANEL)) < 0)
        return -1;
    read_file();
    DisplayPanel (panelHandle);
    RunUserInterface ();
    DiscardPanel (panelHandle);
    return 0;
}
```

Рис. 19. Объявление главной функции, на основе которой строятся все дополнения

Объявление цвета в виде массива позволяет выводить одновременно несколько спектров и окрашивать их в отличающиеся друг от друга цвета. В объявлении «int ColorPlot» нулями обозначены собственные выдуманные цвета предоставленные ниже, и два определенных в среде Lab Windows/ CVI цвета : красный и циан. Подобное объявление позволяет окрашивать каждый график в данные цвета по порядку. Для задания собственного цвета используется функция MakeColor (Red, Green, Blue), где задается интенсивность каждого из трех цветов: красного, зеленного, синего.) Данная функция обозначена на рис.20.

```
int ColorPlot [5]={0,0,0,VAL_RED,VAL_DK_CYAN};

ColorPlot [0]=MakeColor(13,200,223);
ColorPlot [1]=MakeColor(252,155,23);
ColorPlot [2]=MakeColor(234,56,144);
```

Рис.20. Функция обозначения цвета для спектров.

```
int CVICALLBACK Quit (int panel, int control, int event,
        void *callbackData, int eventData1, int eventData2)
{
    switch (event)
    {
        case EVENT_COMMIT:
        QuitUserInterface(0);

            break;
    }
    return 0;
}
```

Рис.21. Функция, отвечающая за работу кнопки «Quit» (выход)

```
int read_file(char fn[])
{
    char s[100];
    FILE *fm;
    int res;

    DisableBreakOnLibraryErrors ();
    fm=fopen (fn, "r");
    EnableBreakOnLibraryErrors ();
    if (fm)


 void ReadIni (void)
 {
     IniText A;
     A = Ini_New(0) ;
     Ini_ReadFromFile (A, "parameters.ini");
     Ini_GetStringIntoBuffer (A, "SetUp", "File name", f, 10);
     Ini_Dispose (A);

 }
```

Рис.22. Начало программы, позволяющее открыть файл измерений с исходным названием, имя файла отправляется в объект № 26 на Рис.12.

```
   fgets(s,99,fm);
       sscanf(s,"%d",&channels);
       SetCtrlVal(panelHandle,PANEL_NUMERIC,channels);
       fgets(s,99,fm);
   sscanf(s,"%d",&startbackground);
       SetCtrlVal(panelHandle,PANEL_NUMERIC_2,startbackground);
         fgets(s,99,fm);
   sscanf(s,"%d",&finishbackground);
       SetCtrlVal(panelHandle,PANEL_NUMERIC_3,finishbackground);
          fgets(s,99,fm);
   sscanf(s,"%g",&angle);
       SetCtrlVal(panelHandle,PANEL_NUMERIC_4,angle);
        fgets(s,99,fm);
        sscanf(s,"%d%d%d",&cycle,&summtime,&monitor );
       SetCtrlVal(panelHandle,PANEL_NUMERIC_5,cycle);
       SetCtrlVal(panelHandle,PANEL_NUMERIC_6,summtime);
       SetCtrlVal(panelHandle,PANEL_NUMERIC_7,monitor);
```

Рис.23. Код построчного чтения открытого файла. Функция «fgets» позволяет считывать строки их файла, «sscanf» - выделять из строки и передавать данные в переменные, функция «SetCtrlVal» - отправлять значение этих переменных в элементы интерфейса пользователя на рабочей панели. Элементы интерфейса пользователя обозначены на Рис.12. под номерами с №1-№17.

```
graph=malloc(channels*sizeof(int));
  for(i=0;i<channels;i++)
  if ((res=fscanf(fm,"%d",&graph[i]))==-1)
      break;

  if (res>0)
      outgraph(waveleight);
  else
      MessagePopup ("Read file", "Error read file");
```

Рис. 24. Функция непосредственно чтения содержимого спектра из файла. При возникновении ошибки недостаточного числа каналов или пустого файла выводится сообщение об ошибке. Строчка «MessagePopup» - выводит на экран текст для пользователя «Error read file» (ошибка чтения файла)

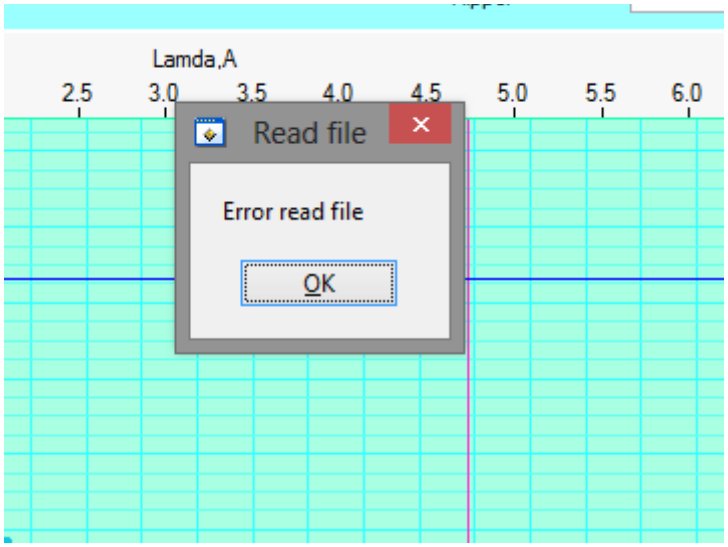


Рис. 25. Демонстрация функции из Рис.13

```
int CVICALLBACK openfile (int panel, int control, int event,
        void *callbackData, int eventData1, int eventData2)
{
    switch (event)
    {
        case EVENT_COMMIT:
        if( FileSelectPopup ("DATA", "*.dat", "", "Open file", VAL_LOAD_BUTTON, 0, 0, 1, 0, fn))
            read_file(fn);

            break;
    }
    return 0;
}
```

Рис.26. Функция кнопки открытия файла с возможностью поиска файлов с расширением «.dat» в корне папки DATA

```
int CVICALLBACK scale (int panel, int control, int event,
        void *callbackData, int eventData1, int eventData2)
{
    switch (event)
    {
        case EVENT_COMMIT:
            int val;
            if(event == EVENT_COMMIT)
            {
                GetCtrlVal(panel, control, &val);
                SetCtrlAttribute (panel,PANEL_GRAPH,ATTR_YMAP_MODE, val);
            }
            break;
    }
    return 0;
}

    double L(void)
    {
        if(flighttime!=0.0)
            return(3.958*wightchannel/flighttime);
        else
            return 0;
    }
```

Рис.27. Функция переключателя обозначенного на Рис.12. № 25. Позволяет переводить график из линейного в логарифмический и обратно, по указанной после объявления функции формуле

```
int outgraph(int wavelenght)
{

    char p[MAX_PATHNAME_LEN];
    char d[MAX_DRIVENAME_LEN];
    char fn2[MAX_FILENAME_LEN];

    // printf("%d\n",plot);
    SetAxisScalingMode (panelHandle, PANEL_GRAPH , VAL_TOP_XAXIS, VAL_MANUAL, 0.0, L()*channels);
    if(plot)
        if (GenericMessagePopup ("Outgraph", "Overlay graphs?", "Yes", "No", "", 0, 0, 0, VAL_GENERIC_POPUP_BTN1, VAL_GENERIC_POPUP_BTN1,
                        VAL_GENERIC_POPUP_NO_CTRL)==2)
    DeleteGraphPlot (panelHandle, PANEL_GRAPH, -1, VAL_IMMEDIATE_DRAW);
    plot=PlotY (panelHandle, PANEL_GRAPH, graph, channels, VAL_INTEGER, VAL_UNSIGNED_INTEGER, VAL_THIN_LINE, VAL_EMPTY_SQUARE, VAL_SOLID, ColorPl


    SplitPath (fn, d, p, fn2);
    SetPlotAttribute (panelHandle, PANEL_GRAPH,plot , ATTR_PLOT_LG_TEXT,fn2);
    readcoursor(panelHandle, PANEL_GRAPH,EVENT_COMMIT,0,0,0);
    return 0;
}
```

Рис. 28. Функция, позволяющая создавать слои из графиков.

В случае, если на графическом окне уже выведен хотя бы один график, запускается Действие функции «GenricMessagePopup». Как и в случае на Рис. 13 будет открываться окно, однако с вопросом «Overlay graphs?» (наложить графики?) и вариантами ответа «Yes» и «No». В случае положительного ответа пользователя, графики накладываются, а отрицательного –графическое окно очищается и выводится только новый график

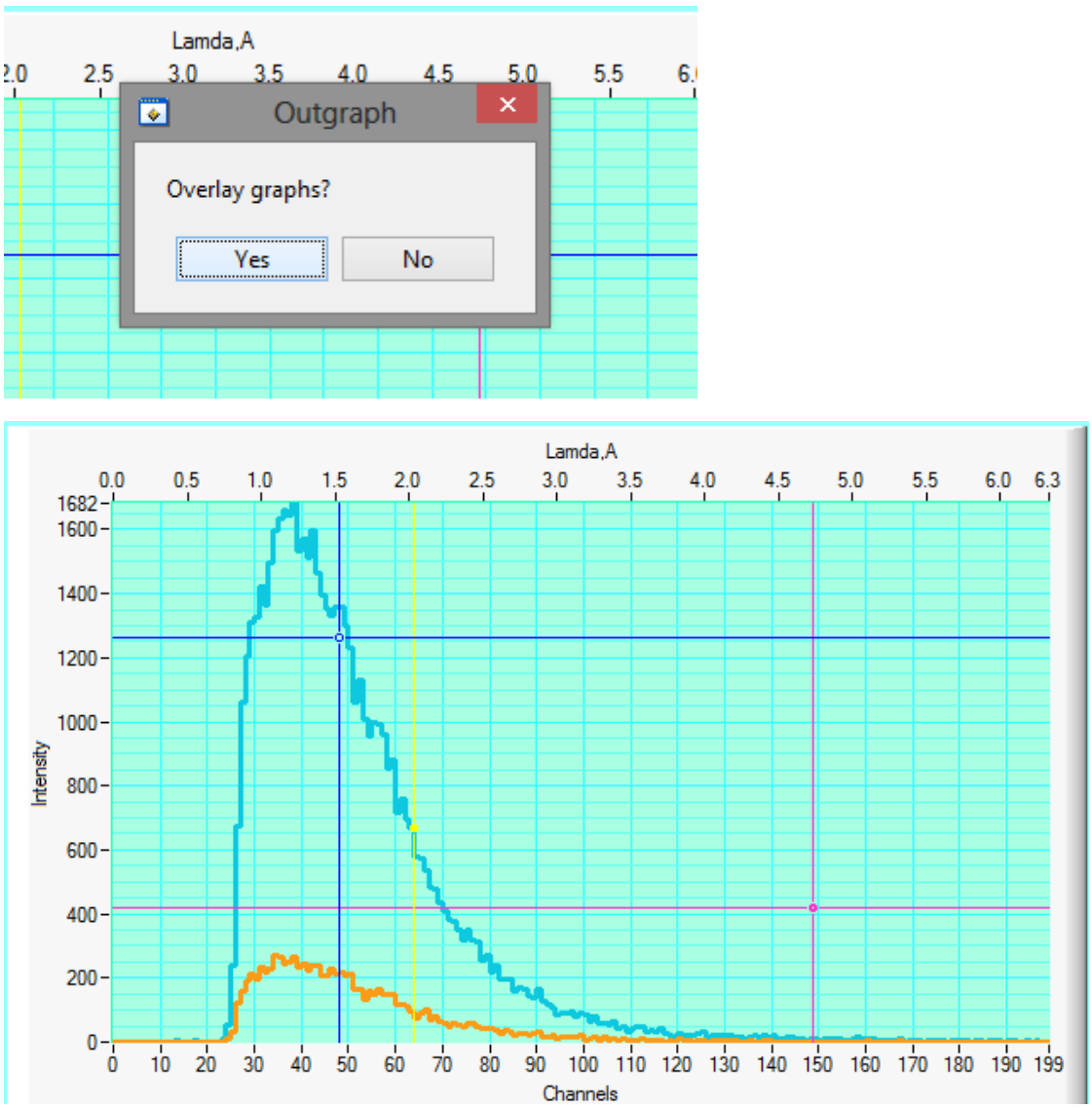


Рис. 29. Демонстрация программы

```
int CVICALLBACK clear (int panel, int control, int event,
        void *callbackData, int eventData1, int eventData2)
{
    switch (event)
    {
        case EVENT_COMMIT:
        DeleteGraphPlot (panelHandle, PANEL_GRAPH, -1, VAL_IMMEDIATE_DRAW);
        plot=0;
            break;
    }
    return 0;
}
```

Рис. 30. Функция кнопки «Clear», очистка от графиков

```
int CVICALLBACK ZoomIn (int panel, int control, int event,
        void *callbackData, int eventData1, int eventData2)
{
    double x1;
    double x2;
    double y1;
    double y2;
    double temp;

    switch (event)
    {
        case EVENT_COMMIT:
        GetGraphCursor(panelHandle, PANEL_GRAPH,1,&x1,&y1);
        GetGraphCursor(panelHandle, PANEL_GRAPH,3,&x2,&y2);

        if(x1>x2)
        {
            temp=x1;
            x1=x2;
            x2=temp;
        }
        if(y1>y2)
        {
            temp =y1;
            y1=y2;
            y2=temp;
        }
        SetAxisRange(panelHandle, PANEL_GRAPH,VAL_MANUAL,x1,x2,VAL_MANUAL,y1,y2);
        SetAxisScalingMode (panelHandle, PANEL_GRAPH , VAL_TOP_XAXIS, VAL_MANUAL, 1.0, 1()*x1, 1()*x2);
```

Рис.31. Функция кнопки Zoom, отвечающая за-масштаб графика, изменение масштаба графика в границах установленными курсорами

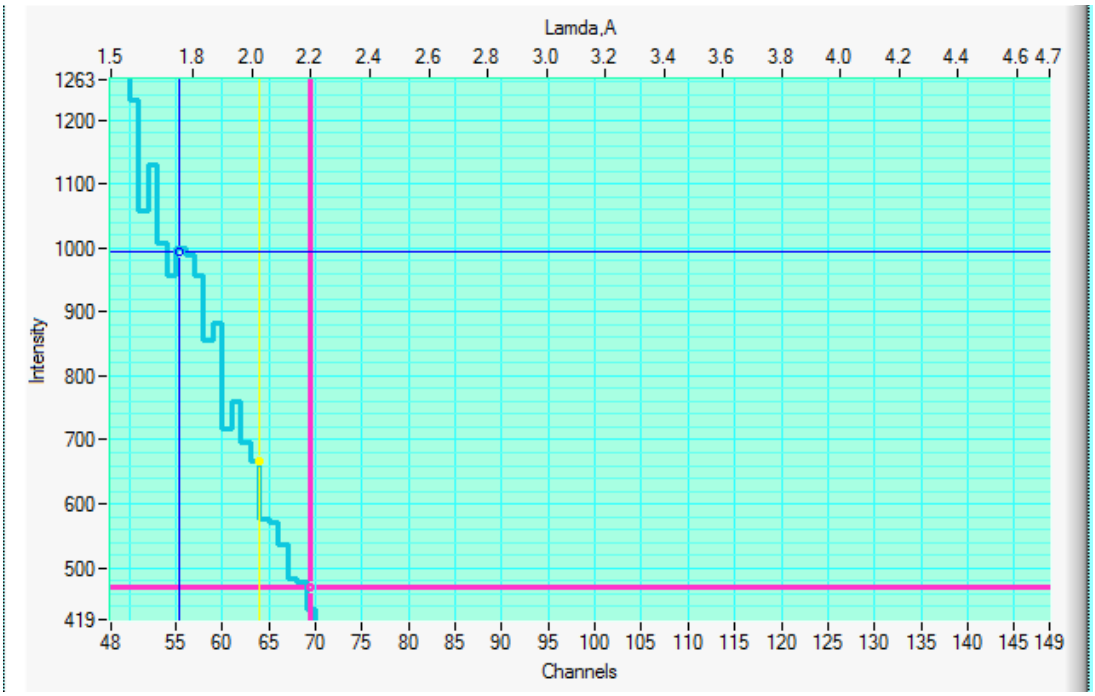


Рис. 32. Демонстрация функции из Рис.31.

```
int CVICALLBACK RestoreGraph (int panel, int control, int event,
        void *callbackData, int eventData1, int eventData2)
{
    switch (event)
    {
        case EVENT_COMMIT:
        SetAxisRange (panelHandle, PANEL_GRAPH, VAL_AUTOSCALE, 0, 0, VAL_AUTOSCALE, 0, 0);
        SetAxisScalingMode (panelHandle, PANEL_GRAPH , VAL_TOP_XAXIS, VAL_MANUAL, 0.0, 1()*channels);

    }
    return 0;
}
```

Рис. 33. Функция кнопки Restore, позволяющая вернуться к автомасштабу

```
int CVICALLBACK readcoursor (int panel, int control, int event,
        void *callbackData, int eventData1, int eventData2)
{
    double x1;
    double y1;
    switch (event)
    {
        case EVENT_COMMIT:
            GetGraphCursor(panelHandle, PANEL_GRAPH,2,&x1,&y1);
            SetCtrlVal(panelHandle, PANEL_NUMERIC_12,y1);
            SetCtrlVal(panelHandle, PANEL_NUMERIC_17,x1);
            SetCtrlVal(panelHandle, PANEL_NUMERIC_18,x1*L());


            break;
    }
    return 0;
}
```

Рис. 34. Функция, с помощью которой можно получать значения (Intensity, Cahnnel, Lambda) любой точки графика в окна вывода данных обозначенные на Рис.12. под номерами 22-24

3.2.5. Графическое представление результатов в программе Spectrum view.

Нижепредставленные графики – это спектры полученные на рефлектометре THP

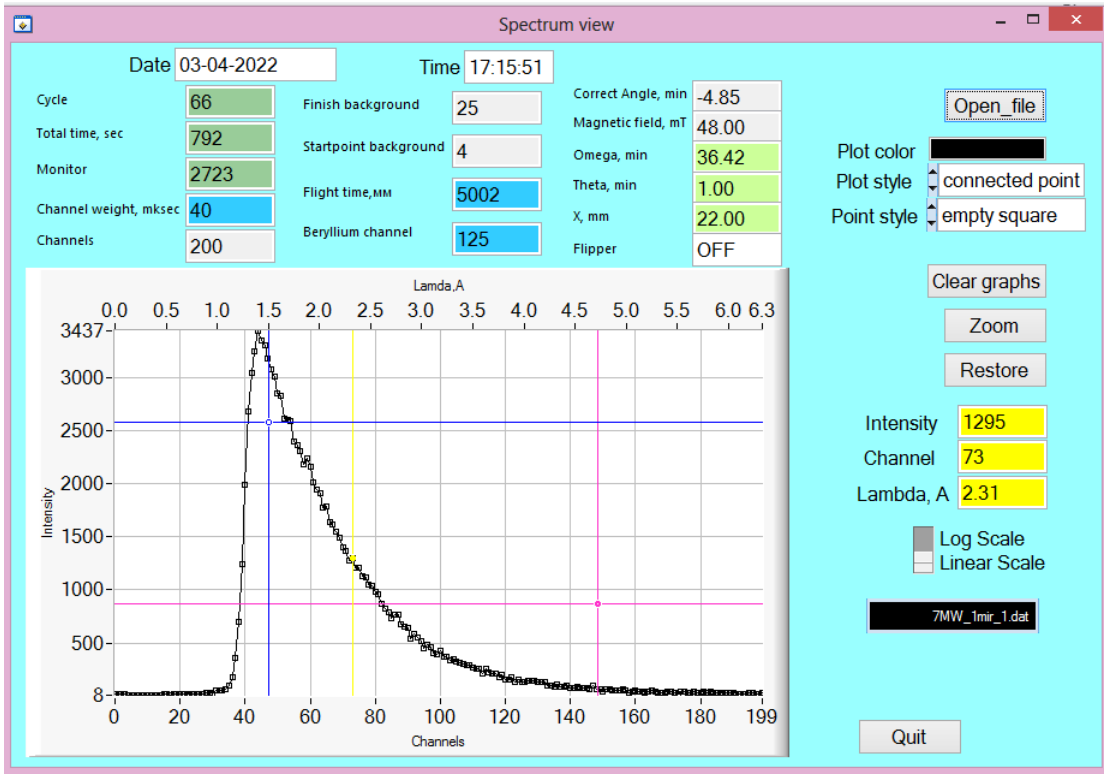


Рис. 35. Спектр на 73 канале с интенсивностью 1295 и длиной волны в 2.31 А

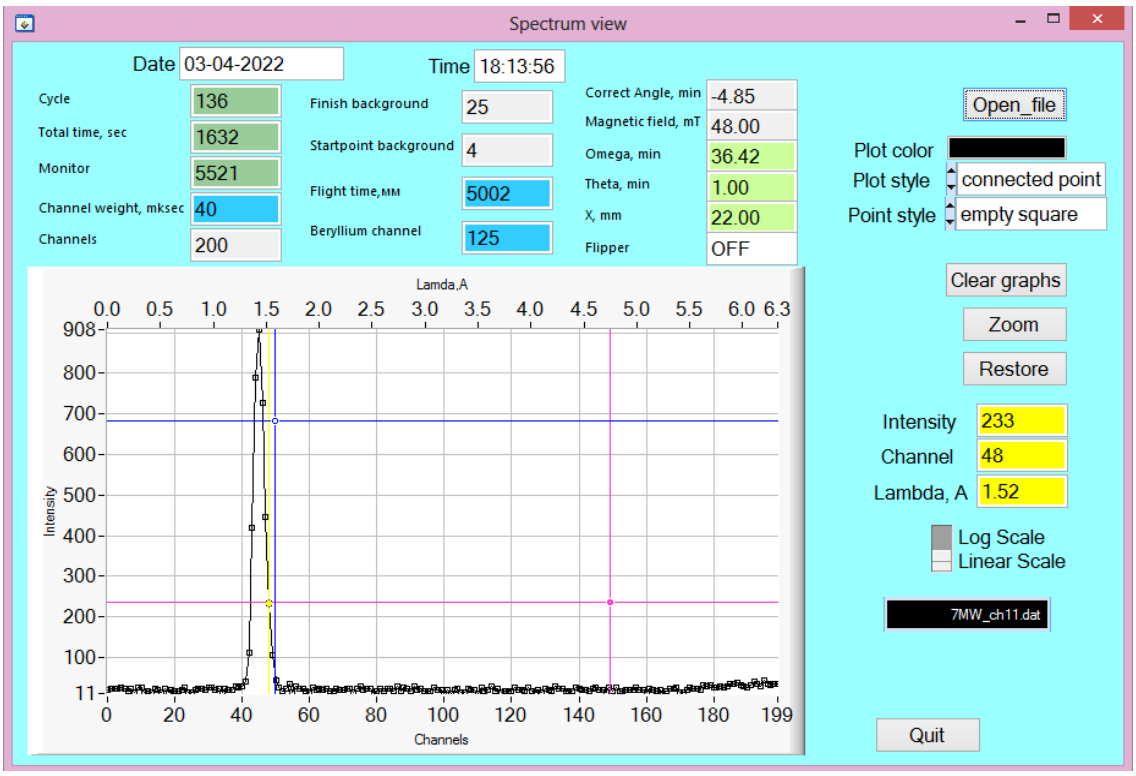


Рис. 36. Спектр на 48 канале с интенсивностью 233 и длиной волны в 1.52 А

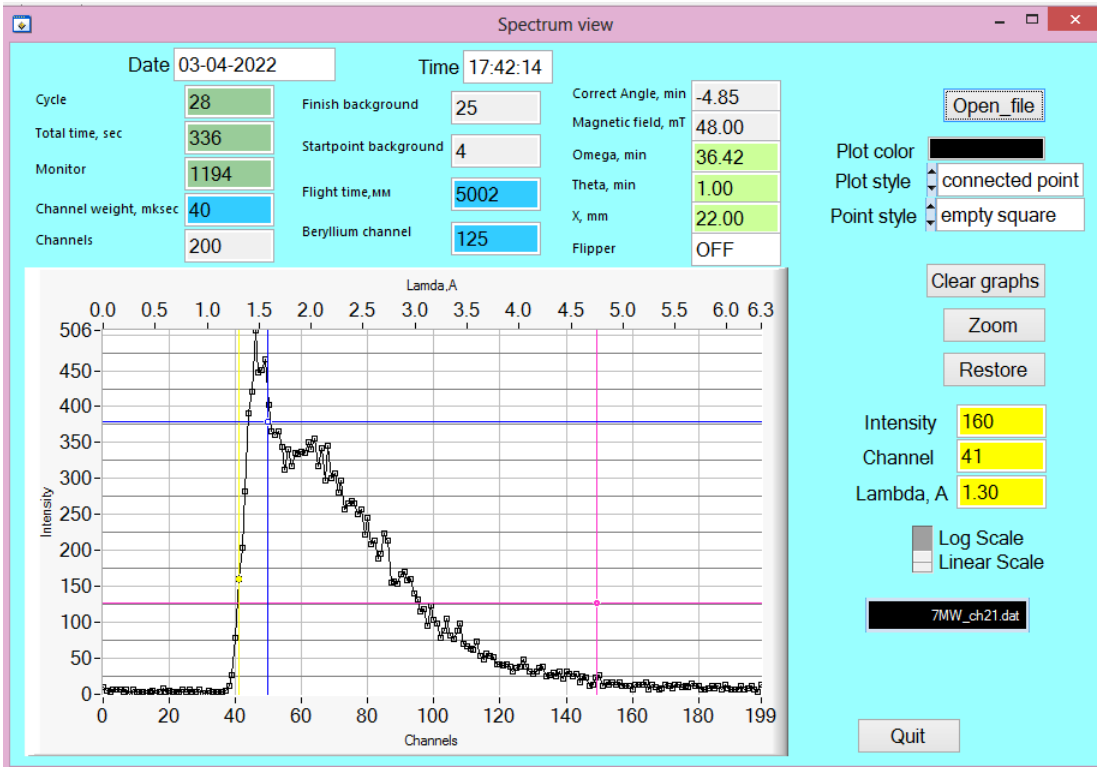


Рис. 37. Спектр на 41 канале с интенсивностью 160 и длиной волны в 1.30 А

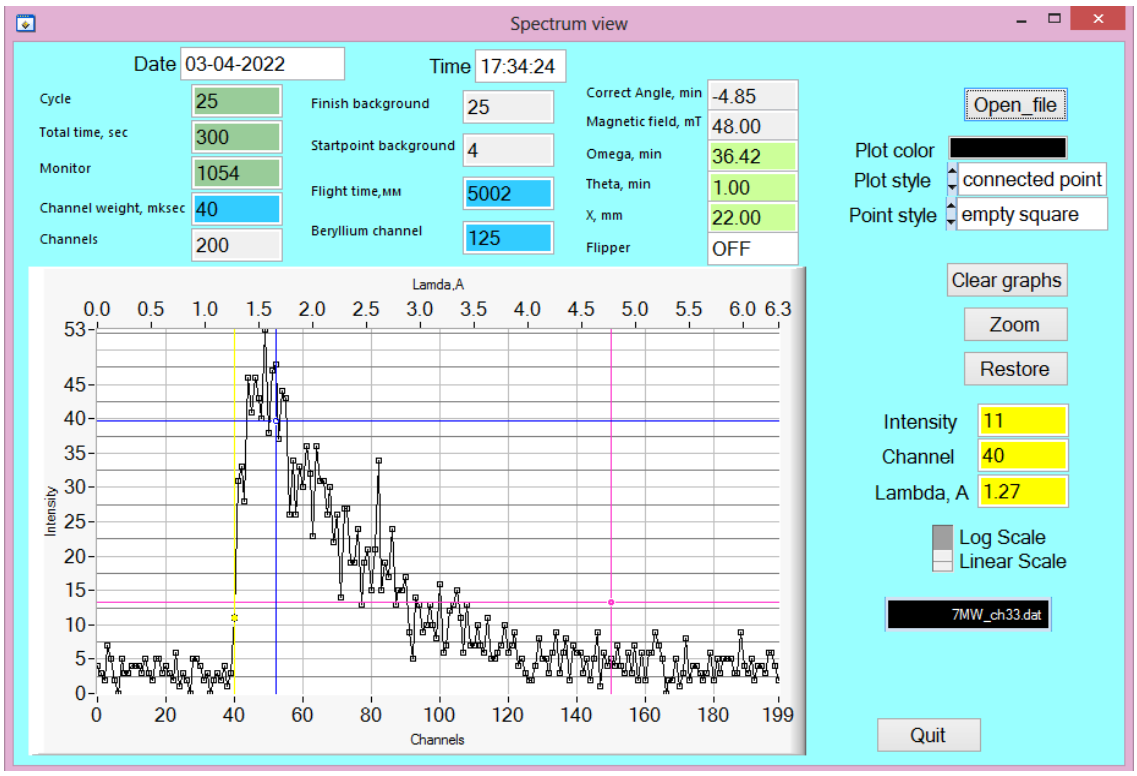


Рис. 38. Спектр на 40 канале с интенсивностью 11 и длиной волны в 1.27 А

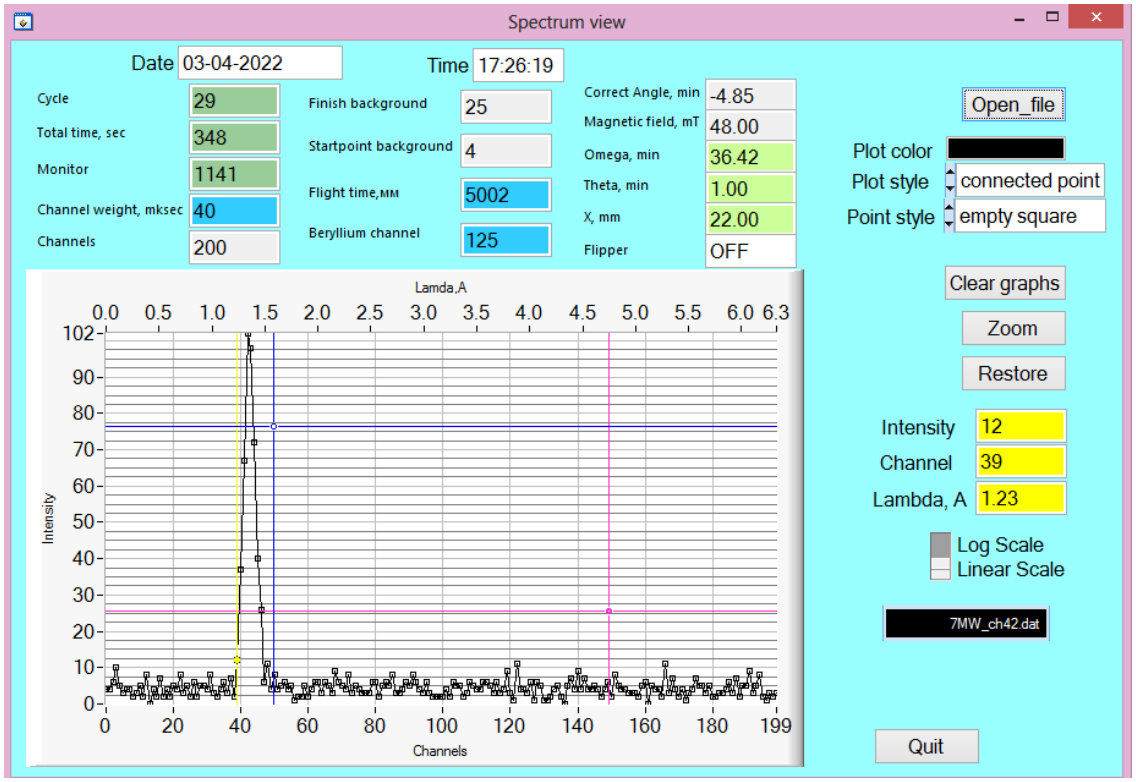


Рис. 39. Спектр на 39 канале с интенсивностью 12 и длиной волны в 1.23 А

Заключение

Целью данной работы было изучение метода исследования образцов многослойных наноструктур с использованием нейтронного пучка, имеющего широкое спектральное распределение методом нейтронной рефлектометрии. Была построена модель с помощью программной среды Mcstas простого рефлектометра с фиксированной длиной волны и реального комбинированного рефлектометра THP – с возможностью переключения режимов от фиксированной длины волны до время-пролетного метода сканирования. А так же осуществлены результаты в виде графических интерпретаций: в первой, ознакомительной модели посредством связи библиотекой iFit с выводом в инженерно-математическую среду Matlab; для второй многоканальной версии была написана программа в среде программирования LabWindows и успешно запущена с полученными данными эксперимента.

Литература

1. Л.Д. Ландау, Е.М. Лившиц. Квантовая механика. - М.: Наука, 1974. - 752 с.
2. Никитенко Ю.В., Сыромятников В.Г. Рефлектометрия поляризованных нейтронов // М.: Физматлит, 2013, 224 с.
3. Гуревич И.И., Тарасов Л.В. Физика нейтронов низких энергий, 1965, 607 с.
4. В.Л. Аксенов. Исследование наносистем и материалов с использованием синхронного излучения и нейтронов в России, 2006.
5. V.G. Syromyatnikov, N.K. Pleshanov, V.M. Pusenkov, A.F. Schebetov, V.A. Ul'yanov, Ya.A. Kasman, S.I. Khakhalin, M.R. Kolkhidashvili, V.N. Slyusar, A.A. Sumbatyan. Four-modes neutron reflectometer NR-4M. - Preprint PNPI № 2619, Gatchina (2005) p.47.
6. LabWindows/CVI [Электронный ресурс] // Engineer Ambitiously: national instruments corp. , 2022, URL: https://www.ni.com/ru-ru/shop/software/products/labwindows-cvi.html#:~:text=LabWindows™%2FCVI%20—%20это%20интегрированная,решения%20задач%20тестирования%20и%20измерения (дата обращения: 20.04.2022).
7. VISIO [Электронный ресурс] // Microsoft, 2022, URL: https://www.microsoft.com/ru-ru/microsoft-365/visio/flowchart-software (дата обращения: 20.04.2022).
8. N.K. Pleshanov, Program UNISON for automatic control of experiments at the reflectometer (1987)
9. В.Г. Сыромятников. Диссертация на соискание ученой степени кандидата физико-математических наук, Гатчина, ПИЯФ (2003).

Приложение.

Приложение 1 . Програмный код модели нейтронного рефлектометра в McStas.

```
DEFINE INSTRUMENT ref(lam_min = 4.95, lam_max = 5.05, lam_w = 5.0, theta_m = 1)

DECLARE
%{
double theta;
double DM_PG = 3.355;
int reflected;
%}

INITIALIZE
%{
theta = asin(lam_w/2/DM_PG)/3.14*180.0;
%}

TRACE

COMPONENT origin = Progress_bar()
AT (0, 0, 0) RELATIVE ABSOLUTE

COMPONENT source_1day = Source_gen(
dist=1.0,
focus_xw=0.005,
focus_yh=0.1,
xwidth = 0.005, yheight = 0.1,    I1=1e13,
T1=30, Lmin = lam_min, Lmax = lam_max)
AT (0, 0, 0) RELATIVE origin
```

```
COMPONENT l_mon1 = L_monitor(
nL=200,
filename="l_mon1.L",
xwidth=0.02,
yheight=0.12,
Lmin=lam_min,
Lmax=lam_max,
restore_neutron=1)
AT (0, 0, 1.01) RELATIVE source_1day

COMPONENT guide = Guide_gravity(
w1=0.005,
h1=0.1,
w2=0.005,
h2=0.1,
l=10.0,
mleft=1,
mright=1,
mtop=2,
mbottom=2)
AT (0, 0, 1.011) RELATIVE source_1day

COMPONENT monok = Monochromator_flat(
zwidth=0.3,
yheight=0.12,
DM=DM_PG,
mosaich = 20.0, mosaicv = 20.0)
AT (0, 0, 10.16) RELATIVE guide
ROTATED (0, -theta, 0) RELATIVE origin

COMPONENT arm = Arm()
```

```
AT (0, 0, 10.16) RELATIVE guide
ROTATED (0, -2*theta, 0) RELATIVE origin

COMPONENT l_mon2 = L_monitor(
nL=200,
filename="l_mon2.L",
xwidth=0.02,
yheight=0.12,
Lmin=lam_min,
Lmax=lam_max,
restore_neutron=1)
AT (0, 0, 0.16) RELATIVE arm


COMPONENT slit1 = Slit(
xwidth = 0.005, yheight = 0.1)
AT (0, 0, 0.17) RELATIVE arm

COMPONENT slit2 = Slit(
xwidth = 0.005, yheight = 0.1)
AT (0, 0, 2.0) RELATIVE slit1
/*
COMPONENT before_mir_xy = Monitor_nD(
xwidth=0.02,
yheight=0.12,
restore_neutron=1,
options="x limits = [-0.01 0.01] bins=25 y limits = [-0.06 0.06] bins = 150")
AT (0, 0, 0.001) RELATIVE slit2

COMPONENT before_mir_dxdy = Monitor_nD(
xwidth=0.02,
```

```
yheight=0.12,
restore_neutron=1,
options="dx limits = [-0.8 0.8] bins=40 dy limits = [-2 2] bins = 100")
AT (0, 0, 0) RELATIVE PREVIOUS
*/
COMPONENT before_mir_x = Monitor_nD(
xwidth=0.008,
yheight=0.12,
restore_neutron=1,
options="x limits = [-0.004 0.004] bins=80")
AT (0, 0, 0.001) RELATIVE slit2

COMPONENT before_mir_dx = Monitor_nD(
xwidth=0.008,
yheight=0.12,
restore_neutron=1,
options="dx limits = [-0.25 0.25] bins=50")
AT (0, 0, 0) RELATIVE PREVIOUS

COMPONENT before_mir_lam = L_monitor(
nL=200,
filename="l_mon3.L",
xwidth=0.008,
yheight=0.12,
Lmin=lam_min,
Lmax=lam_max,
restore_neutron=1)
AT (0, 0, 0) RELATIVE PREVIOUS

COMPONENT mir_start_arm = Arm()
AT (0, 0, 1.01) RELATIVE slit2
```

```
ROTATED (0, 0, 0) RELATIVE arm
/*
COMPONENT mirror = Pol_mirror(
rUpData="supermirror_m3.rfl",
rDownData="supermirror_m3.rfl",
zwidth=2.0,
yheight=0.12, p_reflect = 1)
AT (0, 0, 0) RELATIVE mir_arm
ROTATED (0, theta_m, 0) RELATIVE mir_arm
*/
COMPONENT mirror = Mirror(
reflect="supermirror_m3.rfl",
xwidth=2.0,
yheight=0.12,
center=1,
transmit=1)
AT (0, 0, 0) RELATIVE mir_start_arm
ROTATED (0, 90.0 + theta_m, 0) RELATIVE mir_start_arm
/*EXTEND
%{
if (!SCATTERED) ABSORB;
%}
*/

COMPONENT armReflec = Arm()
AT (0, 0, 0.0) RELATIVE mir_start_arm
ROTATED (0, 2*theta_m, 0) RELATIVE mir_start_arm

COMPONENT armTransmit = Arm()
AT (0, 0, 0.0) RELATIVE mir_start_arm
ROTATED (0, 0, 0) RELATIVE mir_start_arm
```

```
COMPONENT ReflectedBeam = Monitor_nD(
xwidth=0.02, yheight=0.1,
options="x limits = [-0.01 0.01] bins=50", restore_neutron = 1)
AT (0, 0, 1.01) RELATIVE armReflec
GROUP monitorGroup
EXTEND
%{
reflected=1;
%}

COMPONENT TransmittedBeam = Monitor_nD(
xwidth=0.02, yheight=0.1,
options="x limits = [-0.01 0.01] bins=50", restore_neutron = 1)
AT (0, 0, 1.05) RELATIVE armTransmit
GROUP monitorGroup
EXTEND
%{
reflected=0;
%}

COMPONENT R = Monitor_nD(
xwidth=0.02, yheight=0.1,
options="x limits = [-0.01 0.01] bins=50 y limits = [-0.05 0.05] bins=200", restore_neutron = 1)
WHEN (reflected==1) AT (0, 0, 1.01) RELATIVE armReflec

COMPONENT T = Monitor_nD(xwidth=0.02, yheight=0.1,
options="x limits = [-0.01 0.01] bins=50 y limits = [-0.05 0.05] bins=200", restore_neutron = 1)
WHEN (reflected==0) AT (0, 0, 1.05) RELATIVE armTransmit

FINALLY
```

```
%{
%}

END
```

Приложение 2. Spectrum view.h

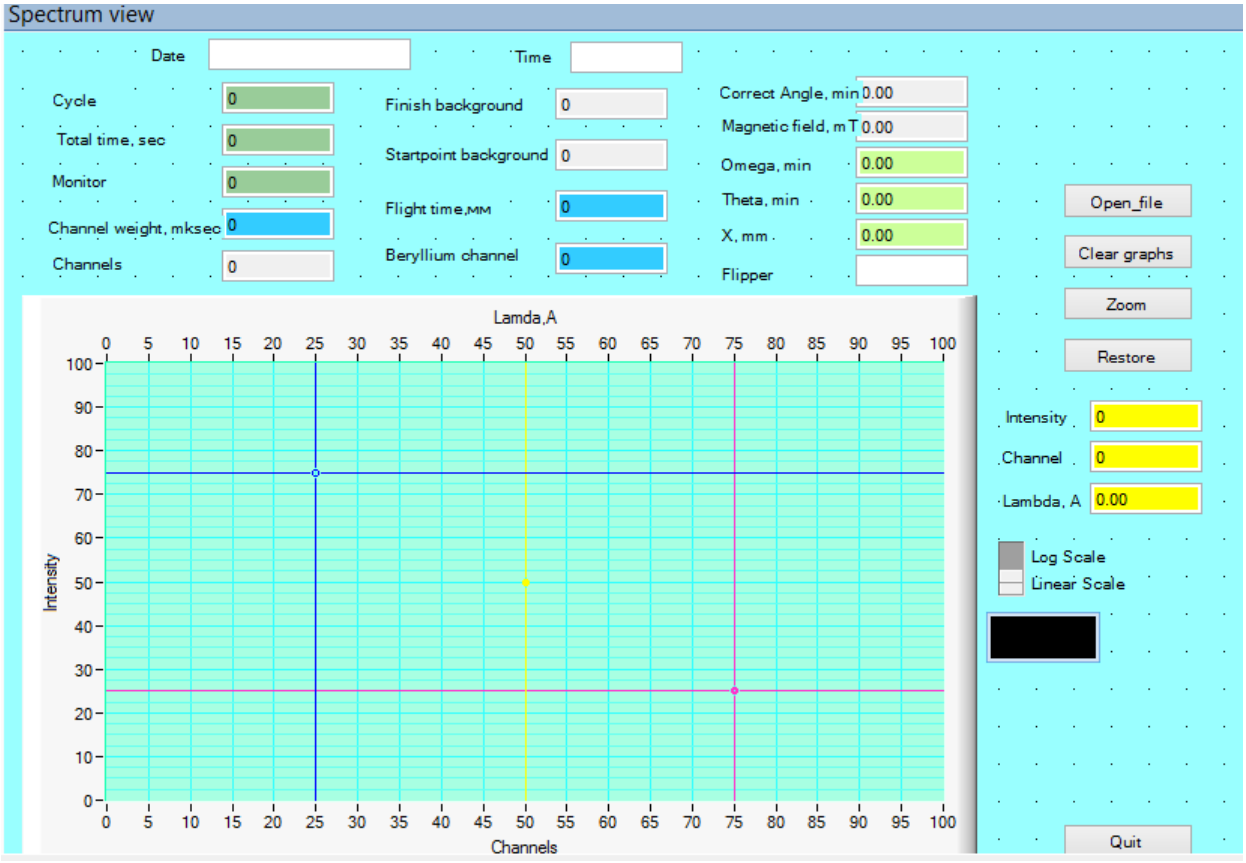
```
/*****************************************************************************/
/* LabWindows/CVI User Interface Resource (UIR) Include File                 */
/* Copyright (c) National Instruments 2021. All Rights Reserved.             */
/*                                                                           */
/* WARNING: Do not add to, delete from, or otherwise modify the contents of  */
/*          this include file.                                               */
/*****************************************************************************/
#include <userint.h>
#ifdef __cplusplus
    extern "C" {
#endif

     /* Panels and Controls: */
#define  PANEL                           1
#define  PANEL_GRAPH                     2       /* control type: graph, callback function:
readcoursor */
#define  PANEL_COMMANDBUTTON             3       /* control type: command, callback
function: Quit */
#define  PANEL_NUMERIC_2                 4       /* control type: numeric, callback function:
(none) */
#define  PANEL_NUMERIC                   5       /* control type: numeric, callback function:
(none) */
#define  PANEL_NUMERIC_3                 6       /* control type: numeric, callback function:
(none) */
#define  PANEL_NUMERIC_4                 7       /* control type: numeric, callback function:
(none) */
#define  PANEL_NUMERIC_5                 8       /* control type: numeric, callback function:
(none) */
#define  PANEL_NUMERIC_6                 9       /* control type: numeric, callback function:
(none) */
#define  PANEL_NUMERIC_7                 10      /* control type: numeric, callback function:
(none) */
#define  PANEL_NUMERIC_8                 11      /* control type: numeric, callback function:
(none) */
#define  PANEL_NUMERIC_9                 12      /* control type: numeric, callback function:
(none) */
#define  PANEL_NUMERIC_10                13      /* control type: numeric, callback function:
(none) */
#define  PANEL_NUMERIC_11                14      /* control type: numeric, callback function:
(none) */
#define  PANEL_NUMERIC_13                15      /* control type: numeric, callback function:
(none) */
```

```
#define  PANEL_NUMERIC_14            16      /* control type: numeric, callback function:
(none) */
#define  PANEL_NUMERIC_15            17      /* control type: numeric, callback function:
(none) */
#define  PANEL_STRING                18     /* control type: string, callback function: (none) */
#define  PANEL_STRING_2              19      /* control type: string, callback function: (none)
*/
#define  PANEL_STRING_3              20      /* control type: string, callback function: (none)
*/
#define  PANEL_COMMANDBUTTON_2       21      /* control type: command, callback
function: openfile */
#define  PANEL_BINARYSWITCH_2        22      /* control type: binary, callback function:
scale */
#define  PANEL_COMMANDBUTTON_3       23      /* control type: command, callback
function: clear */
#define  PANEL_COMMANDBUTTON_4       24      /* control type: command, callback
function: Zoomln */
#define  PANEL_COMMANDBUTTON_5       25      /* control type: command, callback
function: RestoreGraph */
#define  PANEL_NUMERIC_12            26      /* control type: numeric, callback function:
(none) */
#define  PANEL_NUMERIC_18            27      /* control type: numeric, callback function:
(none) */
#define  PANEL_NUMERIC_17            28      /* control type: numeric, callback function:
(none) */


    /* Control Arrays: */
#define  CTRLARRAY                   1
    /* Menu Bars, Menus, and Menu Items: */
        /* (no menu bars in the resource file) */
    /* Callback Prototypes: */
int  CVICALLBACK clear(int panel, int control, int event, void *callbackData, int eventData1,
int eventData2);
int  CVICALLBACK openfile(int panel, int control, int event, void *callbackData, int
eventData1, int eventData2);
int  CVICALLBACK Quit(int panel, int control, int event, void *callbackData, int eventData1,
int eventData2);
int  CVICALLBACK readcoursor(int panel, int control, int event, void *callbackData, int
eventData1, int eventData2);
int  CVICALLBACK RestoreGraph(int panel, int control, int event, void *callbackData, int
eventData1, int eventData2);
int  CVICALLBACK scale(int panel, int control, int event, void *callbackData, int eventData1,
int eventData2);
int  CVICALLBACK Zoomln(int panel, int control, int event, void *callbackData, int
eventData1, int eventData2);
#ifdef __cplusplus
    }
#endif
```

Приложение 3. Spectrum view.uir



Приложение 4. Spectrum view.c

```
#include "inifile.h"
#include <formatio.h>
#include <ansi_c.h>
#include <utility.h>
#include <cvirte.h>
#include <userint.h>
#include "Spectrum view.h"
static int panelHandle;
char f[255], n[255], fn[MAX_PATHNAME_LEN];
int channels;
int startbackground;
int  finishbackground;
float angle;
int cycle;
int summtime;
int monitor;
int Bechannel;
int flighttime;
int wightchannel;
float magneticfield;
char flipper[4];
float omega;
```

```
float theta;
float x;
char date[11];
char endtime[9];
int*graph;
int i;
int waveleight=0;
int plot=0;
int x1;
int y1;
int x2;
int y2;
 int read_file(char fn[]) ;
 int outgraph(int waveleight);
 void ReadIni (void);
 int ColorPlot [5]={0,0,0,VAL_RED,VAL_DK_CYAN};
 int main (int argc, char *argv[])
{if (InitCVIRTE (0, argv, 0) == 0)
return -1;        /* out of memory */
if ((panelHandle = LoadPanel (0, "Spectrum view.uir", PANEL)) < 0)
return -1;
ColorPlot [0]=MakeColor(13,200,223);
ColorPlot [1]=MakeColor(252,155,23);
ColorPlot [2]=MakeColor(234,56,144);
if (argc> 1)
{ ReadIni();
strcpy (n, argv[1]);
Fmt (fn, "DATA\\%s%s.dat", f, n);
read_file(fn);
}
DisplayPanel (panelHandle);
RunUserInterface ();
DiscardPanel (panelHandle);
return 0;
}
int CVICALLBACK Quit (int panel, int control, int event,
void *callbackData, int eventData1, int eventData2)
{switch (event)
{case EVENT_COMMIT:
  QuitUserInterface(0);
break;
}
return 0;
}
int read_file(char fn[])
{ char s[100];
FILE *fm;
int res;
DisableBreakOnLibraryErrors ();
fm=fopen (fn, "r");
EnableBreakOnLibraryErrors ();
if (fm)
```

```
{fgets(s,99,fm);
sscanf(s,"%d",&channels);
SetCtrlVal(panelHandle,PANEL_NUMERIC,channels);
    fgets(s,99,fm);
sscanf(s,"%d",&startbackground);
SetCtrlVal(panelHandle,PANEL_NUMERIC_2,startbackground);
  fgets(s,99,fm);
sscanf(s,"%d",&finishbackground);
SetCtrlVal(panelHandle,PANEL_NUMERIC_3,finishbackground);
   fgets(s,99,fm);
sscanf(s,"%g",&angle);
SetCtrlVal(panelHandle,PANEL_NUMERIC_4,angle);
 fgets(s,99,fm);
 sscanf(s,"%d%d%d",&cycle,&summtime,&monitor );
SetCtrlVal(panelHandle,PANEL_NUMERIC_5,cycle);
SetCtrlVal(panelHandle,PANEL_NUMERIC_6,summtime);
SetCtrlVal(panelHandle,PANEL_NUMERIC_7,monitor);
fgets(s,99,fm);
 sscanf(s,"%d%d%d",&Bechannel,&flighttime,&wightchannel);
SetCtrlVal(panelHandle,PANEL_NUMERIC_8,Bechannel);
SetCtrlVal(panelHandle,PANEL_NUMERIC_9,flighttime);
SetCtrlVal(panelHandle,PANEL_NUMERIC_10,wightchannel);
 fgets(s,99,fm);
sscanf(s,"%g",&magneticfield);
SetCtrlVal(panelHandle,PANEL_NUMERIC_11,magneticfield);
fgets(flipper,4,fm);
SetCtrlVal(panelHandle,PANEL_STRING_2,flipper);
fgets(s,100,fm);
fgets(s,99,fm);
 sscanf(s,"%f%f%f",&omega,&theta,&x);
SetCtrlVal(panelHandle,PANEL_NUMERIC_13,omega);
SetCtrlVal(panelHandle,PANEL_NUMERIC_14,theta);
SetCtrlVal(panelHandle,PANEL_NUMERIC_15,x);
 fgets(date,11,fm);
 SetCtrlVal(panelHandle,PANEL_STRING,date);
   fgets(s,11,fm);
 fgets(endtime,9,fm);
 SetCtrlVal(panelHandle,PANEL_STRING_3,endtime);
 graph=malloc(channels*sizeof(int));
   for(i=0;i<channels;i++)
   if ((res=fscanf(fm,"%d",&graph[i]))==-1)
   break;
   if (res>0)
   outgraph(waveleight);
   else
   MessagePopup ("Read file", "Error read file");
fclose (fm);
  }
return 0;
}
void ReadIni (void)
{
```

```
IniText A;
A = Ini_New(0) ;
Ini_ReadFromFile (A, "parameters.ini");
Ini_GetStringIntoBuffer (A, "SetUp", "File name", f, 10);
Ini_Dispose (A);
}
int CVICALLBACK openfile (int panel, int control, int event,
void *callbackData, int eventData1, int eventData2)
{switch (event)
{case EVENT_COMMIT:
if(       FileSelectPopup ("DATA", "*.dat", "", "Open file", VAL_LOAD_BUTTON, 0, 0, 1, 0,
fn))
read_file(fn);
break;
}
return 0;
}
int CVICALLBACK scale (int panel, int control, int event,
void *callbackData, int eventData1, int eventData2)
{
switch (event)
{
case EVENT_COMMIT:
int val;
if(event == EVENT_COMMIT)
{GetCtrlVal(panel, control, &val);
SetCtrlAttribute (panel,PANEL_GRAPH,ATTR_YMAP_MODE, val);
}
break;
}
return 0;
}
double L(void)
{
if(flighttime!=0.0)
return(3.958*wightchannel/flighttime);
else
return 0;
}
int outgraph(int waveleight)
{
char p[MAX_PATHNAME_LEN];
char d[MAX_DRIVENAME_LEN];
char fn2[MAX_FILENAME_LEN];
// printf("%d\n",plot);
SetAxisScalingMode (panelHandle, PANEL_GRAPH , VAL_TOP_XAXIS, VAL_MANUAL,
0.0, L()*channels);
if(plot)
if (GenericMessagePopup ("Outgraph", "Overlay graphs?", "Yes", "No", "", 0, 0, 0,
VAL_GENERIC_POPUP_BTN1, VAL_GENERIC_POPUP_BTN1,
 VAL_GENERIC_POPUP_NO_CTRL)==2)
DeleteGraphPlot (panelHandle, PANEL_GRAPH, -1, VAL_IMMEDIATE_DRAW);
```

```
plot=PlotY (panelHandle, PANEL_GRAPH, graph, channels, VAL_INTEGER,
VAL_UNSIGNED_INTEGER, VAL_THIN_LINE, VAL_EMPTY_SQUARE, VAL_SOLID,
ColorPlot[plot%5]);
SplitPath (fn, d, p, fn2);
SetPlotAttribute (panelHandle, PANEL_GRAPH,plot , ATTR_PLOT_LG_TEXT,fn2);
readcoursor(panelHandle, PANEL_GRAPH,EVENT_COMMIT,0,0,0);
return 0;
}
int CVICALLBACK clear (int panel, int control, int event,
void *callbackData, int eventData1, int eventData2)
{switch (event)
{case EVENT_COMMIT:
DeleteGraphPlot (panelHandle, PANEL_GRAPH, -1, VAL_IMMEDIATE_DRAW);
plot=0;
break;
}
return 0;
}
int CVICALLBACK Zoomln (int panel, int control, int event,
void *callbackData, int eventData1, int eventData2)
{double x1;
double x2;
double y1;
double y2;
double temp;
switch (event)
{case EVENT_COMMIT:
GetGraphCursor(panelHandle, PANEL_GRAPH,1,&x1,&y1);
GetGraphCursor(panelHandle, PANEL_GRAPH,3,&x2,&y2);
if(x1>x2)
{temp=x1;
x1=x2;
x2=temp;
}
if(y1>y2)
{temp =y1;
y1=y2;
y2=temp;
}
SetAxisRange(panelHandle, PANEL_GRAPH,VAL_MANUAL,x1,x2,VAL_MANUAL,y1,y2);
SetAxisScalingMode (panelHandle, PANEL_GRAPH , VAL_TOP_XAXIS, VAL_MANUAL,
L()*x1, L()*x2);
}
return 0;
}
int CVICALLBACK RestoreGraph (int panel, int control, int event,
void *callbackData, int eventData1, int eventData2)
{switch (event)
{case EVENT_COMMIT:
SetAxisRange (panelHandle, PANEL_GRAPH, VAL_AUTOSCALE, 0, 0,
VAL_AUTOSCALE, 0, 0);
```

```
SetAxisScalingMode (panelHandle, PANEL_GRAPH , VAL_TOP_XAXIS, VAL_MANUAL,
0.0, L()*channels);
}
return 0;
}
int CVICALLBACK readcoursor (int panel, int control, int event,
void *callbackData, int eventData1, int eventData2)
{double x1;
double y1;
switch (event)
{case EVENT_COMMIT:
GetGraphCursor(panelHandle, PANEL_GRAPH,2,&x1,&y1);
SetCtrlVal(panelHandle, PANEL_NUMERIC_12,y1);
SetCtrlVal(panelHandle, PANEL_NUMERIC_17,x1);
SetCtrlVal(panelHandle, PANEL_NUMERIC_18,x1*L());
break;
}
return 0;
}
```