Письма в ЭЧАЯ. 2016. Т. 13, № 1(199). С. 208–219

НОВЫЕ ВЕРСИИ ПРОГРАММ ЮСТИРОВКИ И ВИЗУАЛИЗАЦИИ СПЕКТРОВ ДЛЯ РЕФЛЕКТОМЕТРОВ РЕАКТОРА ИБР-2

А. С. Кирилов [1]

Объединенный институт ядерных исследований, Дубна

Работа посвящена основным возможностям программ юстировки рефлектометров (ICE) и визуализации спектров (SpectraViewer), используемых на рефлектометрах реактора ИБР-2 ЛНФ ОИЯИ. Программа ICE реализована как надстройка над управляющим комплексом Sonix+ и предназначена для предварительной настройки спектрометра перед основным измерением. Программа SpectraViewer применяется также и на других спектрометрах. Программы написаны с использованием PyQt и графической библиотеки matplotlib.

The article is devoted to the main features of the tuning program (ICE) and the spectra visualization program (SpectraViewer), used at reflectometers of the IBR-2, FLNP, JINR. The ICE program is implemented as an add-on for the control software Sonix+ and is designed to adjust the instrument before the main measurement. The SpectraViewer program is also used at other instruments. Programs have been written using the PyQt and the graphics library matplotlib.

PACS: 07.05.-t

ВВЕДЕНИЕ

Рефлектометры — группа спектрометров для проведения измерений с поляризованными нейтронами, находящихся на 8-м (РЕМУР), 9-м (РЕФЛЕКС) и 10-м (ГРЕЙНС) пучках реактора ИБР-2 [1]. Каждое измерение, как правило, предваряется процедурой настройки (юстировки) спектрометра для поиска отраженного пучка. Обычно для этого проводится одно- или двумерное сканирование по выбранной оси (осям) с измерением в каждой точке и последующим анализом полученных спектров для определения оптимального положения. Для автоматизации этого процесса на спектрометре РЕМУР В. Е. Юдиным в 2003 г. была разработана программа ICE (Integrated Calibration Environment) [2]. Программа была выполнена как надстройка над комплексом Sonix (позднее Sonix+) [3], написана на языке Visual C++ и была рассчитана строго на имеющийся состав оборудования спектрометра. Особенностью программы также было то, что визуализация данных была реализована автором без использования каких-либо графических библиотек. Позднее часть кода программы ICE, отвечающая за визуализацию данных,

[1] E-mail: akirilov@nf.jinr.ru

была оформлена в виде самостоятельной программы SpectraViewer. В целом, программа оказалась очень удачной и успешно использовалась до остановки реактора в 2006 г.

Модернизация спектрометра, в частности замена одномерного позиционно-чувствительного детектора (1D ПЧД) на двумерный (2D ПЧД), сделала использование варианта В. Е. Юдина невозможным, поскольку такой тип детектора программой не поддерживался. Кроме того, возникла необходимость адаптации ICE для других рефлектометров (РЕФЛЕКС и ГРЕЙНС). Полная переработка программы была выполнена С. Велешки [4]. Перед ним была поставлена задача, используя вариант В. Е. Юдина как прототип, предложить и реализовать общий подход к построению программ юстировки с целью упрощения адаптации программы к различному составу оборудования спектрометров, измерительным процедурам и алгоритмам оптимизации.

В новой версии преемственность в интерфейсе программы была сохранена, при этом:

— программа обеспечивала организацию его как набора компонент (модулей) для облегчения построения разных вариантов;

— в программу была добавлена вариативность в задании списка осей сканирования с помощью XML-файлов;

— процедура измерения в точке вынесена в отдельный скрипт (файл на Python);

— для визуализации спектров и измеряемых зависимостей использована библиотека matplotlib [5];

— для реализации GUI использован PyQT [6].

Таким образом были подготовлены версии для спектрометров РЕМУР, РЕФЛЕКС, ЮМО, «Эпсилон», ДН-6 и ДН-12, причем в двух последних случаях программа используется в качестве основного пользовательского интерфейса.

К сожалению, С. Велешки не смог завершить свою работу, и ее продолжение было передано автору данной статьи. Анализ кода показал необходимость еще одной итерации в разработке программы (см. разд. «Реализация»).

Были добавлены следующие возможности (первые три варианта возможностей присутствовали в варианте В. Е. Юдина, но отсутствовали в варианте С. Велешки):

— сканирование «R-отношений» (отношений результатов двух измерений с разной поляризацией);

— сканирование по двум осям (двойной вложенный цикл);

— вычисление позиции средневзвешенного значения спектра и переход в нее, а также переход в позиции максимума и минимума спектра;

— сканирование по детекторам всех типов (монодетекторы, 1D и 2D ПЧД).

Также появились новые сервисные возможности:

— визуализация результатов двумерного сканирования в виде карты плотности;

— автоматическое добавление точек в график зависимостей, сопровождаемое звуковым сигналом;

— запись графика зависимости в формате ASCII-файла.

Интерфейс программы был упрощен, а компонента визуализации спектров (SpectraViewer) обновлена (см. ниже «Визуализация спектров»).

ОСНОВНЫЕ ВОЗМОЖНОСТИ ПРОГРАММЫ

Как было отмечено выше, в новой версии сохранена высокая преемственность с вариантом В. Е. Юдина и в функциональности, и в интерфейсе.

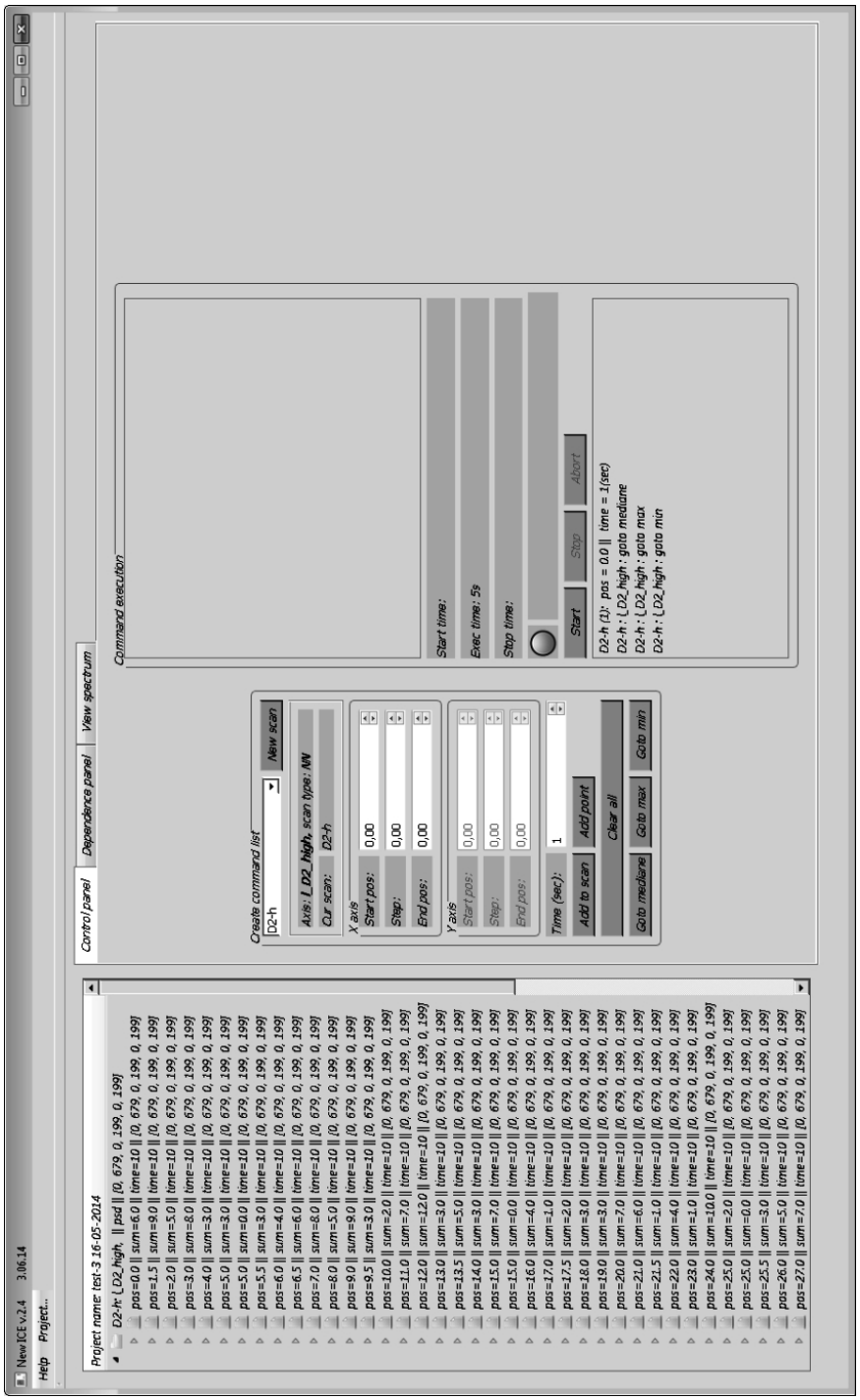

Рис. 1. Окно навигации и контрольная панель

Измерения группируются в проекты, которые, в свою очередь, состоят из сканов — последовательностей измерений в отдельных точках. Есть возможность открывать проекты, удалять их или вводить новые. Аналогичные операции есть для сканов и точек.

На рис. 1 слева расположено окно навигации. В нем содержится информация о текущем проекте:

— имя проекта,

— сканы и соответствующие точки в виде дерева.

Каждая точка содержит ссылку на один или два спектра.

Контрольная панель позволяет выбирать скан, создавать новый скан либо добавлять дополнительные точки к уже имеющимся. Вновь созданные точки добавляются в нижний список (список на исполнение). По мере выполнения точки переходят в верхний список. Текущая точка показана между списками. С помощью кнопок «Start», «Stop» и «Abort» пользователь может управлять процессом сканирования.

Данные о текущем проекте и содержимое списка на исполнение сохраняются и автоматически восстанавливаются при вызове программы.

При задании нового скана (рис. 2) пользователь вводит его имя и выбирает требуемые значения параметров скана:

- оси сканирования,
- тип сканирования,
- детектор и границы диапазона суммирования.

На вкладке «Dependence panel» (рис. 3) отображаются зависимости для одномерных сканов в графической и табличной формах. Текущий скан, т. е. скан, измерение которого продолжается или закончено последним, изображается кривой *1* и отображается в таблице справа. Для сравнения на график можно поместить одну или две кривые, соответствующие другим сканам проекта или загрузить их из ASCII-файла.

Галочка «Auto scaling» задает режим автоматического масштабирования для графиков. При ее снятии кривые масштабируются в соответствии со значениями $X_{\min}$, $X_{\max}$, $Y_{\min}$, $Y_{\max}$.

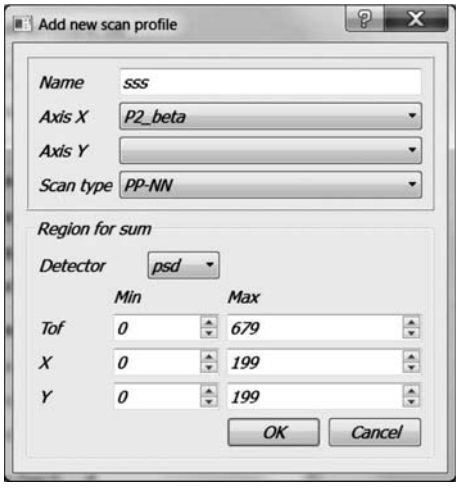

Рис. 2. Задание нового скана

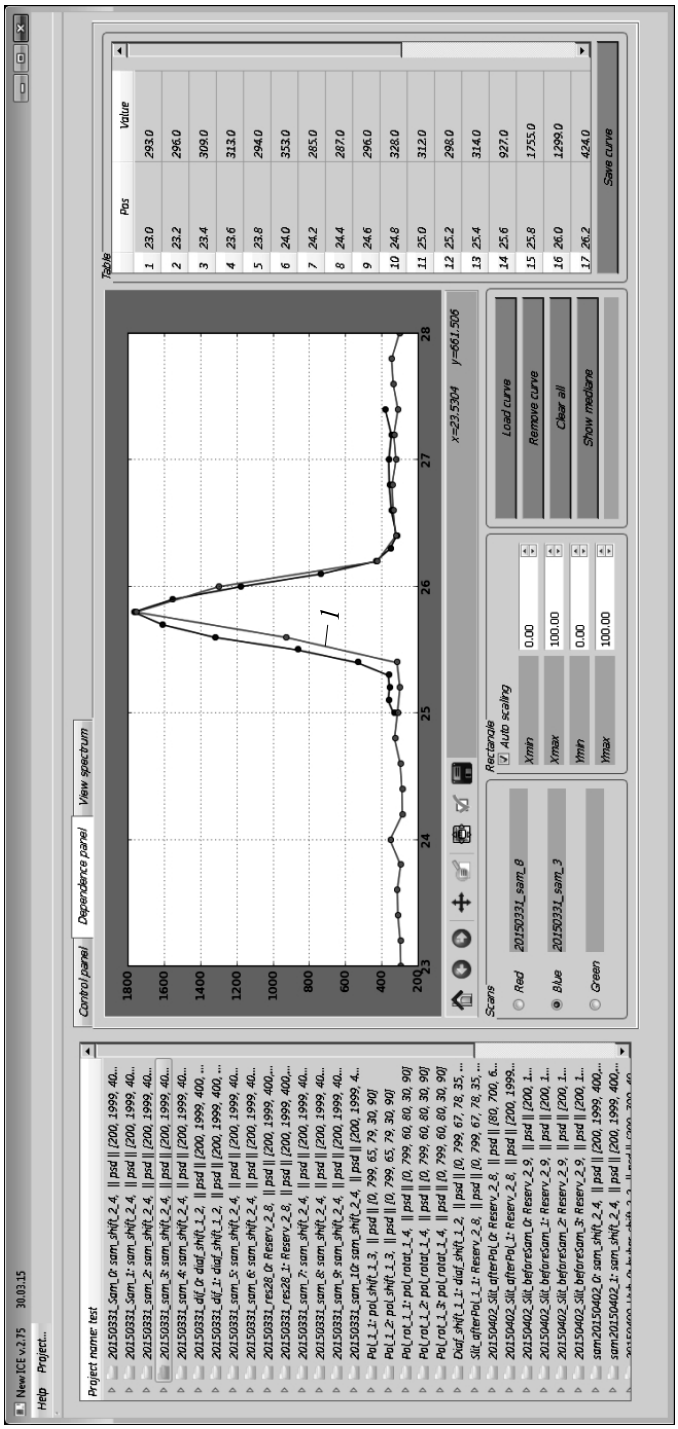

Рис. 3. Визуализация зависимостей

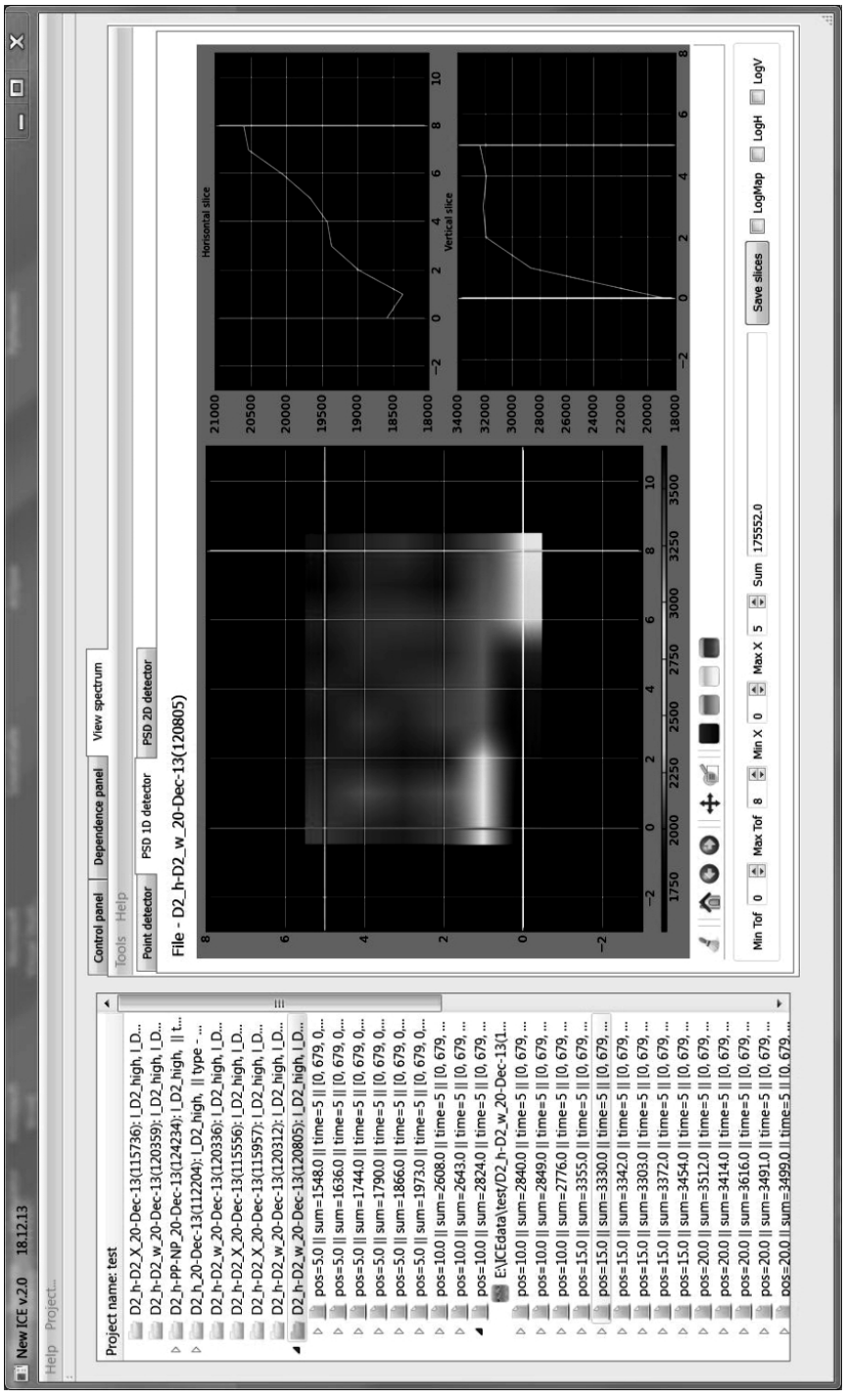

Рис. 4. Изображение зависимости для двумерного сканирования

Кнопка «Show mediane» предназначена для вычисления позиции равновесного сечения площади кривой. Медиана изображается вертикальной пунктирной линией.

Визуализация зависимостей, полученных при двумерном сканировании, изображается в виде карты плотности как данные с одномерного ПЧД (рис. 4). При этом доступны все операции для анализа, предусмотренные соответствующим виджетом (см. «Визуализация спектров»).

РЕАЛИЗАЦИЯ

После детального знакомства с вариантом С. Велешки было принято решение упростить реализацию, отказавшись в некоторой степени от универсальности. В этом варианте специфические для каждого спектрометра операции были собраны в отдельном модуле «медиатора», т. е. посредника. Это, в целом правильное, решение было реализовано настолько сложно, что очень затрудняло понимание внутренней логики работы программы и, соответственно, ее коррекцию.

Очередная переработка программы значительно как упростила ее внутреннюю структуру, так и сократила ее объем. В целом, число модулей сократилось втрое, а их суммарный объем более чем вчетверо.

В частности, были удалены:

— модуль медиатора, затрудняющего понимание внутренней логики работы программы;

— все операции с XML-файлами, они заменены на прямое сохранение/восстановление внутренних структур с помощью модуля pickle;

— виртуальные классы — за ненадобностью;

— модули описания графического интерфейса, построенные с помощью QtDesigner'а.

XML-файлы использовались для конфигурирования параметров, например, списка осей, а также для сохранения/восстановления состояния программы между запусками. Это решение было и нерациональным, и неудобным. Во-первых, в Sonix+ все параметры установки уже хранятся в так называемой «базе данных» (БД) Varman [7] и дополнительные списки не нужны. Во-вторых, содержимое XML-файлов при запуске программы копировалось во внутренние структуры ICE. При изменении этих значений нужно было обновлять содержимое XML-файлов. Для чтения/записи файлов были составлены специальные модули, которые при изменении структуры параметров также нуждались в корректировке. С другой стороны, использование модуля pickle позволяет сохранять/восстанавливать структуру любой сложности с помощью одной команды, делая применение XML бессмысленным.

В варианте С. Велешки программа ICE для выполнения измерения в каждой точке записывала параметры измерения в несколько переменных БД, далее через интерпретатор Sonix+ запускала специально созданный скрипт и ожидала его окончания. Такая схема позволяла простую реализацию возможностей приостановки и прекращения измерений. Эта схема взаимодействия ICE с программой измерения в точке была сохранена, но сам протокол взаимодействия изменен. В новом варианте scanICE2 задание записывается в одну переменную ICEcommand, а об окончании измерения сигнализирует запись результата в преременную ICEstatus (см. приложения).

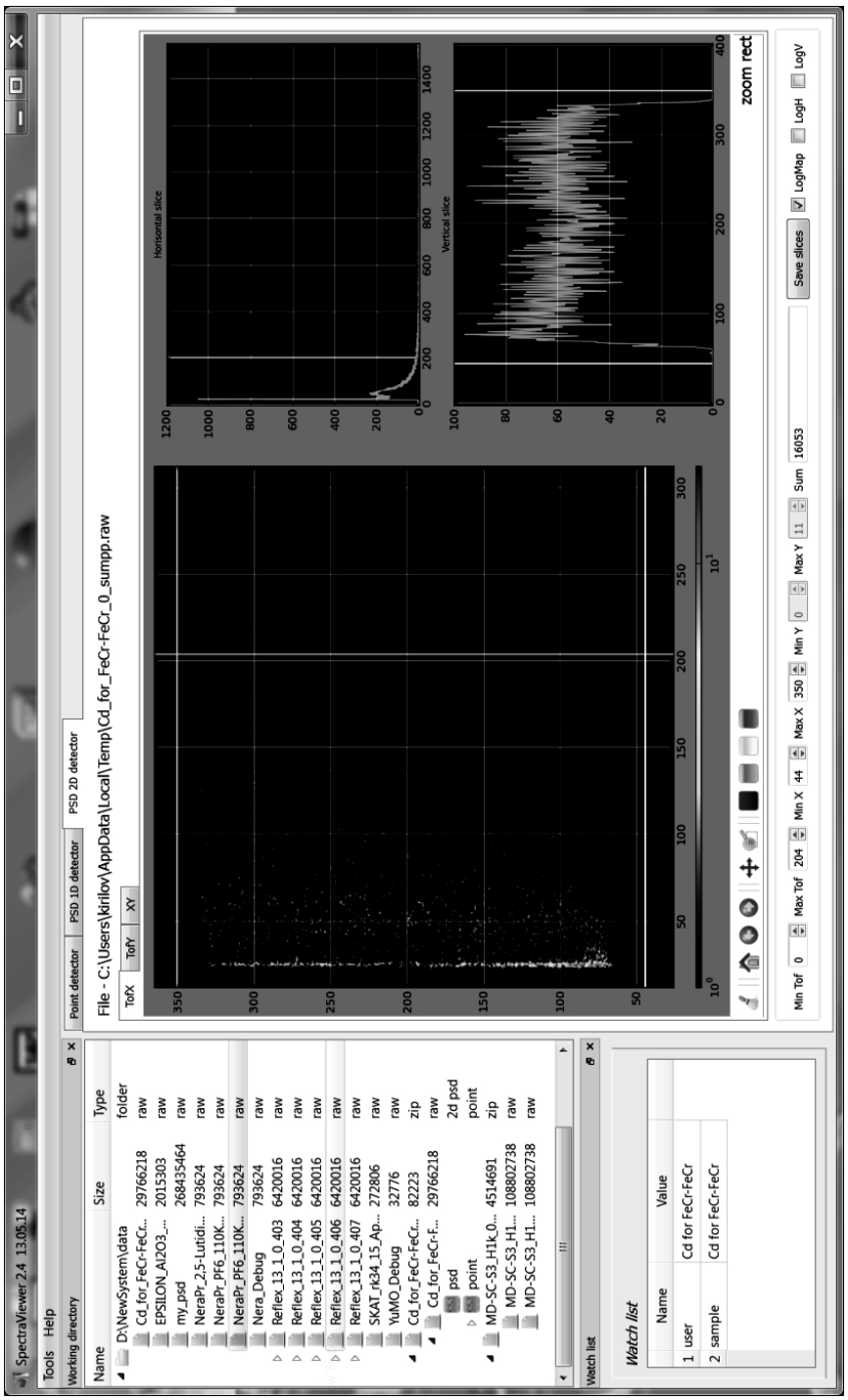

Рис. 5. Представления карты плотности для данных, полученных с 2D ПЧД

Кроме того, в библиотеку специфических операций для спектрометра (ICE_lib2) вынесены операции получения возможных списков (см. приложение): осей, типов сканирования, операция для подсчета времени измерения в точке, а также операции подсчета значений для графика зависимостей, поскольку формула подсчета зависит от типа сканирования.

Выделены программы расчета в отдельный модуль, чтобы избежать ошибки с повторным вызовом NumPy [8].

ВИЗУАЛИЗАЦИЯ СПЕКТРОВ

Для визуализации спектров используется программа SpectraViewer (вкладка View spectrum). Это та же программа, которая используется как самостоятельно, так и в качестве компонента вместе с другими виджетами в новом GUI [9].

Вариант, подготовленный С. Велешки, также подвергся значительным переделкам, для повышения надежности, сокращения времени построения графиков и расширения функциональных возможностей. Наиболее значимые из них:

— в окно навигации добавлены операции для работы с упакованными файлами;

— для одномерных данных введены представление в логарифмического масштабе и отключение автоматического масштабирования с заданием границ окна представления;

— для двумерных данных введено раздельное «логарифмирование» карты плотности и срезов, по просьбе пользователей изменен фон и карта цветов, осуществлена замена фона и «карт»;

— добавлено окно для просмотра параметров измерения из БД Varman.

Реализация представления трехмерных данных была полностью переработана с использованием виджетов представления двумерных данных (рис. 5). Дополнительно был подготовлен вариант для работы в режиме offline, в котором необходимая информация извлекается не из БД Varman, а из файла, содержащего ее образ.

ЗАКЛЮЧЕНИЕ

В настоящее время программа юстировки успешно эксплуатируется на всех рефлектометрах, а программа визуализации SpectraViewer также и на других спектрометрах. Вместе с тем, уже проясняются возможности для дальнейшего усовершенствования, выдвигаются новые требования, следовательно, работа над программой будет продолжена.

Автор выражает благодарность А. В. Петренко за активную позицию при формулировании задания и терпение при отладке программы, В. И. Приходько за ценные критические замечания при подготовке данной рукописи. Автор признателен всем коллегам за помощь в работе и поддержку.

Приложение 1
ПРОГРАММА СКАНИРОВАНИЯ В ТОЧКЕ (scanICE2.py)

Программа предназначена для измерения в каждой точке последовательности сканирования (режим «point») либо для перехода в точку «медианы» (режим «goto»). Специфика рефлектометра содержится в именах библиотеки, файла конфигурации и командах управления моторами (осями).

```
#    ICE point execution procedure for Remur instrument
import comm_dll
from remur_python_configuration import *
import remur_lib
import sys
import protocol_dll
from aux_scan_operations2 import *

res = comm_dll.connect('Test')
if res !='OK':
    protocol_dll.WriteErrorToProtocol( 'comm_dll connect error')
    sys.exit(-1)

try:
    command = comm_dll.GetVar('ICEcommand')
#
#command = "I_D1_width|2.0|PN|1|D:/NewSystem/ICE_data/my_ice_test/test/
  ICE6|psd| [10, 100, 0, 199, 0, 199]"
#
    protocol_dll.WriteToProtocol( 'ICE command:'+ command)
                                          # parse command
    cl = command.split('|')
    code = cl[0]
    if code == 'point':
        axis_x = cl[1]
        pos_x = float(cl[2])
        scan_type = cl[3]
        expo_time = int(cl[4])
        file_path = cl[5]
        det_name = cl[6]
        sumr = GetSumr(cl[7])
        nDim = int(cl[8])
        if nDim == 2:
            axis_y = cl[9]
            pos_y = float(cl[10])
            motor_y = eval(axis_y)
            res = motor_y.Set(pos_y)
            protocol_dll.WriteToProtocol( 'motor_y command:'+ str(res))
            ErrorExit(res)

        motor_x = eval(axis_x)
        res = motor_x.Set(pos_x)
        protocol_dll.WriteToProtocol( 'motor_x command:'+ str(res))
        ErrorExit(res)

        if scan_type in single_scan_list:
            fname = file_path + '_' + scan_type+ '.raw'
            SingleMeasurement(scan_type, expo_time, fname)
            res = '0'+'| 0.0 |'+fname
            protocol_dll.WriteToProtocol(res)
```

```
        elif scan_type in double_scan_list:
            type = scan_type[:2]
            fname1 = file_path + '_' + type+ '.raw'
            SingleMeasurement(type, expo_time, fname1)

            type = scan_type[3:]
            fname2 = file_path + '_' + type+ '.raw'
            SingleMeasurement(type, 2*expo_time, fname2)
            res = '0'+'| 0.0 |'+fname1+'|'+fname2
            protocol_dll.WriteToProtocol(res)
        else:
            res = '1'+'|'+'Bad scan type'
    elif code == 'goto':
        axis_x = cl[1]
        pos_x = float(cl[2])
        motor_x = eval(axis_x)
        res = motor_x.Set(pos_x)
        protocol_dll.WriteToProtocol( 'motor_x command:'+ str(res))
        ErrorExit(res)
        res = '0|goto'
        else:
            res = '' # fatal error

    finally:
        print 'finally...', res
        if res == '':
            protocol_dll.WriteErrorToProtocol( 'Fatal error')
            res = '2|Fatal error'
        comm_dll.SetVar('ICEstatus', res)
        if res[0] != '0':
            protocol_dll.WriteErrorToProtocol(res)
        else:
            protocol_dll.WriteToProtocol(res)
```

Приложение 2
СПИСОК ДОПОЛНИТЕЛЬНЫХ ОПЕРАЦИЙ (aux_scan_operations2.py)

Дополнительные операции определяются спецификой рефлектометра. Это списки режимов сканирования (single_scan_list, double_scan_list), собственно процедура «элементарного» измерения (SingleMeasurement (type, expo_time, fname)), процедура подсчета времени измерения в точке (getRealExpoTime (type, time)), которое определяется временем работы двигателя, установленными задержками и проч., а также процедура суммирования измеренных спектров в каждой точке (GetSumr(s)). Из-за особенностей NumPy эту процедуру нельзя было включить в программу ScanICE2, и она вызывается в ICE непосредственно.

Приложение 3
БИБЛИОТЕКА ОПЕРАЦИЙ СКАНИРОВАНИЯ (ICE_lib2.py)

В этой библиотеке для удобства собраны различные операции, которые используются в ICE для получения информации о детекторах и других устройствах в Sonix+, а также операции по суммированию сегментов спектров различной размерности. Все эти операции со спецификой рефлектометров не связаны.

```
def RecalculateSum(scan_type, file_list, det_name, data_type, det_range, new_range,
det_num = 0, det_type = 0)
def SectorSum(fname, det_name, data_type, det_range, new_range, det_num, det_type)
def SectorSum2D(fname, det_name, data_type, det_range, new_range)
def SectorSum1D(fname, det_name, data_type, det_range, new_range)
def SectorSumMono(fname, det_name, data_type, det_range, new_range, det_num)
def GetDeviceList(server_name)
def GetAxesList()
def GetScanTypeList()
def GetActiveDetectorsList()
def GetDetectorRanges(detector)
def GetDectectorType(detector)
def _GetNumberOfDetectorsInSet(detector)
def GetDetectorFullInfo(detector)
def GetPositionAxis(nameAxis)
```

СПИСОК ЛИТЕРАТУРЫ

1. http://flnp.jinr.ru/558/
2. *Юдин В. Е.* Программа юстировки спектрометра РЕМУР в среде MS Windows. Сообщ. ОИЯИ Р13-2003-12. Дубна, 2003. 10 с.
3. http://sonix.jinr.ru/wiki/doku.php?id=ru:index
4. *Белушкин А. В. и др.* Многосекционный кольцевой детектор тепловых нейтронов для исследования дифракции на микрообразцах в аксиальной геометрии // Письма в ЭЧАЯ. 2013. Т. 10, № 5(182). С. 713–721.
5. http://matplotlib.org/
6. http://www.riverbankcomputing.co.uk/software/pyqt/download
7. *Кирилов А. С., Юдин В. Е.* Реализация базы данных реального времени для управления экспериментом в среде MS Windows. Сообщ. ОИЯИ Р13-2003-11. Дубна, 2003. 7 с.
8. http://www.numpy.org/
9. *Kirilov A. S. et al.* The Unified GUI for Neutron Instrument Control Based on PyQt // Proc. of the NEC'2013. Dubna: JINR, 2013. P. 158–160.

Получено 22 мая 2015 г.