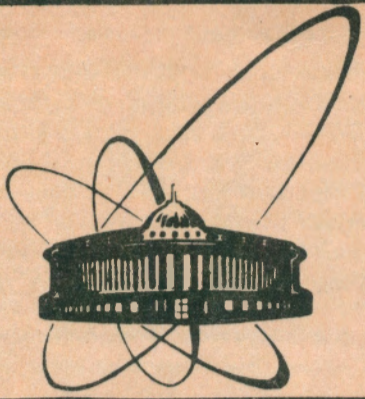

сообщения
объединенного
института
ядерных
исследований
дубна

P11-92-416

О.Г.Смирнова, Т.А.Стриж, В.Э.Файн

ИСПОЛЬЗОВАНИЕ СИСТЕМЫ PAW В MS DOS

1992

1. ВВЕДЕНИЕ

Рабочее место для физического анализа[1] Physics Analysis Workstation (PAW)– это универсальная система для интерактивного анализа данных, которая объединяет различные системы, пакеты и инструментальные средства для управления данными и представления информации, созданные в течение последних 15 лет в ведущих физических центрах мира, в основном для работ в физике высоких энергий. Система[2] обеспечивает автоматизацию всех этапов научной работы, связанной с анализом экспериментальной и другой физической информации (включая информацию, получаемую в ходе модельных расчетов на ЭВМ):

- хранение и анализ больших объемов информации на устройствах прямого доступа;
- их графическое представление;
- статистический и математический анализ;
- преобразование;
- обмен данными, включая передачу информации по системам электронной почты, EtherNet или на магнитных лентах, дисках, дискетах и т.п.;
- использование одновременно нескольких процессоров или связанных в единую сеть ЭВМ, если для решения задачи необходимо увеличение вычислительной мощности;
- подготовку к публикации графических материалов, оформление стендов, плакатов, слайдов для публичных выступлений

и т.д.

Система не требует от исследователя специальной подготовки. Пользователю предоставляется специальный объектно-ориентированный язык, объектами которого являются такие хорошо понятные для естествоиспытателя вещи, как гистограммы, статистические распределения, графики, рисунки, вектора и скаляры. Развитый аппарат макросредств помогает вводить свои собственные команды, объекты и операции.

В особо сложных случаях имеется возможность воспользоваться таким хорошо известным физикам языком программирования, как ФОРТРАН.

К сожалению, до последнего времени система PAW была практически недоступна многим исследователям, т.к. была реализована для суперЭВМ, "больших" машин и дорогих рабочий станций (типа SUN, Appolo, HP и т.д.)[3]. Появление дешевых и мощных IBM PC-совместимых машин с параметрами, близкими к параметрам рабочих станций (типа IBM PC AT 386/486), адаптация в ОИЯИ[4, 5] для этих машин системы PAW и включение версии системы для персональных ЭВМ в стандартный РАМ-файл ЦЕРН[6] открыло новые возможности для широкого использования этой современной и мощной системы не только

в физике высоких энергий, но и в других областях, где требуется физический анализ экспериментальной информации, а также в высших учебных заведениях.

Данная публикация не ставит своей целью дать точное и подробное изложение системы PAW и ее подсистем. Авторы хотели дать самое общее представление о системе, особенностях ее реализации для персональных ЭВМ типа IBM PC AT386/486 с операционной системой MS DOS. Приводится несколько "живых" примеров, демонстрирующих уникальные возможности системы, для того чтобы исследователь смог оценить их и принять решение об использовании PAW в своих задачах и необходимости освоения системы в полном объеме. Здесь мы старались продемонстрировать только те возможности системы, которые не нашли достаточного отражения в "Руководстве для пользователей"[1], но, как показал наш опыт, весьма полезных на практике.

2. ТЕРМИНЫ PAW

Некоторые термины, используемые в описании системы PAW[1] и в данном препринте, возможно, являются не вполне устоявшимися и непривычны для пользователей, не работающих в области физики высоких энергий. Ниже мы приводим термины на английском языке и их перевод, используемые в данной публикации[1] и литературе, на которую мы ссылаемся в тексте.

Histogram – Гистограмма – это одно- или двумерный массив данных, сгенерированных пакетом HBOOK в пакетном режиме или в ходе работы с пакетом PAW. *Гистограмма* (явно или неявно) описывается (заказывается), заполняется явно заданными данными или содержимым другой *гистограммы*. Информация о *гистограмме* включает в себя: заголовок (title), определение каналов и способа их упаковки, содержимое каналов (bin) и ошибки (errors), статистические параметры, возможно, функциональный вектор и выходные атрибуты. Собранная вместе вся эта информация и составляет *гистограмму*;

Booking – Заказ – операция **описания** (создания) *гистограммы*;

Filling – Заполнение – операция **ввода** данных в заданную *гистограмму*;

Fitting – Фитирование – определение параметров *функции*, наилучшим образом описывающей заданную *гистограмму* или *вектор* методом наименьших квадратов или максимального правдоподобия;

Projection – Проекция – операция проектирования двумерного распределения на обе оси;

Band – Лента – это *проекция* на ось X (или Y) заданного интервала вдоль другой оси Y (или X);

[1] В тексте, там, где это необходимо, мы будем выделять определенные ниже термины *курсивом*.

Slice – Срез – это *проекция* на ось X (или Y) выбранного интервала размером в один канал вдоль другой оси Y (или X). Таким образом, *срез* является особым случаем *ленты*, у которой размер интервала ограничен размерами одного канала;

Ntuple – N-банк – реляционная **двумерная таблица**[2], состоящая из **фиксированного** числа (N) столбцов, определяющего количество входов на элемент (например, это могут быть координаты N-мерного фазового пространства изучаемого физического явления) и произвольного числа строк, определяемого общим числом (**длиной** *N-банка*) элементов в *N-банке* (например, числом измеренных точек в N-мерном фазовом пространстве). Каждый элемент *N-банка* можно рассматривать как описание независимого физического события. Определяя критерии выборки элемента из *N-банка*, можно быстро и эффективно анализировать как отдельные элементы, так и содержимое всего *N-банка* целиком в диалоговом режиме;

DST – ЛСР или ФСР – Data Summary Tape – Лента Суммарных Результатов или Data Summary File – Файл Суммарных Результатов — **набор** экспериментальных данных с последовательной организацией;

Event – Событие – подмножество данных в *ЛСР*, относящихся к отдельному физическому событию;

Cut – Выборка – функция выборки, используемая при анализе *N-банка*;

Mask – Маска – файл, содержащий для каждого элемента заданного *N-банка* признаки – удовлетворяет ли этот элемент указанному набору условий в результате применения к нему критерия *выборки*;

Macro – Макро – текстовый файл, содержащий команды системы PAW и логические операторы для управления порядком их выполнения. *Макро* может иметь параметры, значения которых определяются в момент вызова *Макро*;

Vector – Вектор – эквивалент понятия **массив** в языке ФОРТРАН. Допускаются, максимально, 3-мерные *вектора*. Элементом *вектора* могут быть как вещественные, так и целые числа. Они могут вводиться в диалоговом режиме с клавиатуры или считываться из внешнего файла;

Metafile – Метафайл – файл, содержащий графическую информацию в формате, независимом от конкретного графического устройства;

Picture – Рисунок – графический объект, состоящий из графических примитивов и атрибутов. *Рисунки* генерируются графической подсистемой HIGZ и могут быть сохранены в базе данных рисунков, построенной с помощью RZ-пакета системы управления структурами – ZEBRA;

[2]Так как нам не удалось подобрать удачного русского оквивалента термину Ntuple, в этой публикации мы будем использовать термин **N-банк**.

PostScript – широко распространенный **язык** описания страниц, используемый для управления лазерными принтерами;

Function – Функция – последовательность "ФОРТРАН-подобных" **операторов**, вводимых в интерактивном режиме из командной строки или внешнего файла.

3. СТРУКТУРА ПАКЕТА PAW

Пакет PAW построен на базе и объединяет ряд инструментальных средств и пакетов, каждый из которых может быть использован самостоятельно. Многие из них хорошо известны, широко используются в научных исследованиях и имеют свою историю (HBOOK[7] и HPLOT[8], SIGMA[1], MINUIT[9]). Назначение PAW — помочь физикам в анализе и визуализации их данных. Он обеспечивает графическое представление, статистический и математический анализ информации. Пакет может использоваться специалистами, имеющими только самые общие сведения об ЭВМ. Для того, чтобы помочь новичку, пакет имеет широкие возможности вывода на экран подробных инструкций о своей работе и перехода, в случае необходимости, в режим меню. С другой стороны, наличие мощного аппарата макрокоманд и встроенного интерпретатора с языка ФОРТРАН (COMIS), делают пакет PAW мощным и эффективным инструментом в руках опытного программиста. На рис.1[3] дана общая схема взаимодействия отдельных компонент системы.

3.1. KUIP – инструмент для создания интерфейсов

Пакет KUIP[10] (**K**it for a **U**ser **I**nterface **P**ackage) – это инструментальная система для создания на ФОРТРАНе прикладного программного обеспечения с "дружественным" интерфейсом. Он необходим для поддержания диалогового взаимодействия между пользователем и прикладной программой (в нашем случае – PAW). В функции пакета входит организация ввода и анализ командной строки, проверка введенных параметров на корректность и передача управления исполнительной подпрограмме.

Синтаксис команд, воспринимаемых KUIP, определяется с помощью **Command Definition File** (CDF – файл определения команд) и информацией, зафиксированной в ZEBRA-структуре, которая используется не только на стадии синтаксического анализа, но также и в тех случаях, когда пользователь в ходе диалога просит систему о помощи (**online help**) и KUIP "сам" формирует необходимую команду. Все команды упорядочены в виде древовидной структуры и могут при вводе сокращаться до любого допускающего однозначное толкование числа символов. Если введенной строки не хватает для однозначного распознавания команды, KUIP подскажет все возможные варианты. С помощью аппарата

[3]Рисунок взят из [1] и публикуется с разрешения R.Brun.

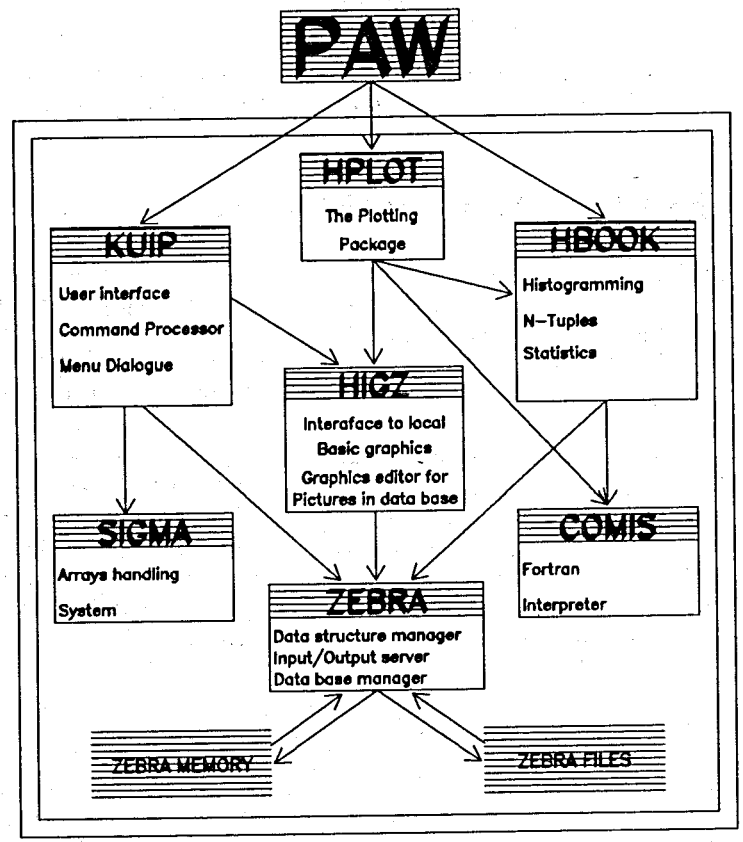

Рис. 1: Схема взаимодействия отдельных компонент системы PAW

синонимов (**Alias**) пользователь имеет возможность вводить для часто повторяемых команд и параметров свои собственные сокращения. Последовательность команд PAW может быть зафиксирована в текстовом файле, который, кроме команд системы PAW, может содержать управляющие операторы (`IF-THEN-ELSE; CASE; DO-WHILE; GOTO` и т.п.). Таким образом, в руках исследователя имеется мощный *макроязык*, который позволяет программировать решение достаточно общих задач анализа и обработки данных в интерактивном режиме (Точно так же, как это можно делать, например, с помощью ФОРТРАНа). Пользователь может самостоятельно выбирать различные стили ведения диалога (командные и меню) и переходить с одного способа ведения диалога на другой в любой момент времени. Для того, чтобы сэкономить время, при вводе большинства команд можно использовать принцип указания значений параметров "по умолчанию". В специальном системном стеке пакета KUIP (**history file**) постоянно хранится последовательность из N последних введенных команд, которые могут быть в

любой момент просмотрены, скопированы, отредактированы и исполнены. Содержимое стека предыдущего и текущего сеансов работы с PAW сохраняются на диске.

3.2. HBOOK и HPLOT — пакеты для работы с гистограммами

HBOOK и его графический интерфейс HPLOT — это ФОРТРАН-библиотеки для работы с гистограммами, их статистического анализа и графического представления, которые используются уже много лет. Они обеспечивают:

- работу с одно- и двумерными *гистограммами* и *N-банками*;
- работу с *проекциями* и *срезами* двумерных *гистограмм* и *N-банков*;
- широкие возможности управлять порядком заполнения *гистограмм* и извлечения содержимого *гистограмм*;
- операции над *гистограммами*, сравнение *гистограмм*;
- инструменты для минимизации и параметризации;
- генерацию случайных чисел с произвольной заданной исследователем функцией распределения
- создание базы данных *гистограмм* и *N-банков* в памяти;
- сохранение *гистограмм* и *N-банков* в файлах прямого доступа пакета ZEBRA;
- широкий набор графических возможностей:
  - обычные гистограммы, столбчатые гистограммы, объемные гистограммы, нанесение коридора ошибок, круговые и цветные гистограммы;
  - построение гладких кривых и поверхностей;
  - построение диаграмм рассеяний, двумерных гистограмм, изолиний и трехмерных поверхностей;
  - автоматическое создание графических "окон";
  - ввод координат с помощью графического указателя (например, манипулятора "мышь", "светового" пистолета и т.п.)

3.3. HIGZ – подсистема для создания, представления, хранения и модификации рисунков

HIGZ — интерфейс высокого уровня для создания, представления, хранения и модификации графиков и рисунков (High level Interface to Graphics and Zebra) был разработан как составная часть проекта PAW. Пакет представляет собой промежуточный слой между прикладной программой (например, PAW) и базовым графическим пакетом (например, графической библиотекой транслятора NDP-ФОРТРАН на PC или пакетом GKS). Кроме непосредственного вывода графических объектов с помощью средств машинной графики, он обеспечивает:

- создание базы данных рисунков и графиков;
- полную переносимость графической базы данных (база данных рисунков);
- простую манипуляцию с элементами рисунков;
- максимально компактное представление графических данных, обеспечивающее их передачу и обработку и хранение в файлах прямого доступа;
- независимость прикладных программ от используемого базового графического пакета. В настоящее время HIGZ может работать на базе нескольких версий GKS, X windows, GL (Silicon Graphics), GDDM (IBM), GPR, GMR3D (Appolo), DI3000, на машинах IBM PC с любым типом графического монитора (CGA, EGA, VGA и т.п.).

HIGZ обеспечивает все вышеперечисленные возможности благодаря использованию системы управления данными — ZEBRA.

Разработка HIGZ проводилась так, чтобы его базовые понятия были как можно ближе к стандарту GKS[11]. В пакете HIGZ не вводится каких-либо новых базовых графических возможностей, но в него встроены некоторые наиболее часто используемые на практике графические макропримитивы, такие как дуги, оси координат, прямоугольники, круговые диаграммы, таблицы и т.д.

Система позволяет:

- использовать базовые графические функции и макропримитивы высокого уровня;
- использовать локальную графическую подсистему; при этом имена подпрограмм и их параметры мало отличаются от соответствующих функций стандарта GKS;
- управлять структурами данных, используя интерфейс с пакетом ZEBRA;
- обеспечивать интерактивное создание и редактирование рисунков.

Все эти возможности доступны одновременно, что крайне необходимо при интерактивной работе, т.к. пользователь имеет возможность оперативно реагировать и немедленно исправлять ранее созданные рисунки, без повторного исполнения прикладной программы.

Система обеспечивает получение рисунков на языке *PostScript* и LaTeX, что значительно облегчает пользователю публикацию своих результатов.

3.4. ZEBRA — система управления структурами данных

Система управления структурами данных ZEBRA была разработана в ЦЕРН для того, чтобы компенсировать отсутствие возможностей работы с динамическими структурами данных на языке ФОРТРАН, основном языке программирования, используемом в физике высоких энергий. Пакет обеспечивает динамическое создание и модификацию структур данных в ходе работы программы, обмен с внешними запоминающими устройствами как на том же самом компьютере, так и на другой ЭВМ (может быть даже ЭВМ совершенно другого типа), передачу данных из одной области памяти в другую, взаимодействие с другими компьютерами в сети ЭВМ с минимальными накладными расходами по времени работы.

ZEBRA может поддерживать любые типы структур, но имеет специальные средства для поддержки линейных и древовидных структур (списков). Ввод/вывод данных может осуществляться как на устройства прямого, так и последовательного доступа. Имеется возможность воспользоваться двумя способами представления информации: естественным (**native**) и транспортным (**exchange**). В первом случае при записи/чтении/передаче информации никаких преобразований не происходит, т.е. все числа и символы хранятся на внешних устройствах во внутреннем представлении той ЭВМ, на которой в данный момент времени работает ZEBRA. Во втором предполагается, что в ходе записи информации все данные были преобразованы к специальному стандартному формату передачи и обмена информацией. В этом случае структуры данных могут свободно передаваться с машины на машину как одной и той же, так и совершенно разной архитектуры без дополнительной перекодировки, что особенно важно в физике, где, как правило, сбор и первичная обработка информации осуществляется персональными и миниЭВМ, а окончательный их анализ и интерпретация — на рабочих станциях и суперЭВМ.

3.5. MINUIT — система минимизации и анализа ошибок

MINUIT — это инструмент для нахождения экстремума многопараметрической функции, анализа типа экстремума и поведения функции в окрестности точки экстремума. Пакет может быть использован для статистического анализа качества аппроксимации по критерию χ^2 и методом максимального правдоподобия, а также для определения соответствующих оптимальных параметров. В сложных, плохо обусловленных случаях исследователь имеет возможность управлять процессом решения "вручную", фиксировать отдельные параметры, удалять или добавлять отдельные экспериментальные точки, изменять величину погрешности измерений, шаг аппроксимации и т.п.

3.6. COMIS — ФОРТРАН-интерпретатор

Интерпретатор COMIS позволяет пользователю интерактивно как вводить, так и выполнять набор подпрограмм на языке ФОРТРАН. Интерпретатор реализует практически полностью стандартную версию языка ФОРТРАН-77. Это очень важный инструмент, так как позволяет исследователю определять свои собственные алгоритмы анализа данных в сложных случаях, например, свою собственную целевую (аппроксимирующую) *функцию* в задаче минимизации.

3.7. SIGMA — язык обработки массивов

Язык программирования – SIGMA (System for Interactive Graphical Mathematical Applications) был разработан для автоматизации расчетов на ЭВМ математиками и физиками-теоретиками и используется в ЦЕРН и других ядерно-физических центрах уже более 10 лет. Включен в PAW.

Его основные характеристики:

- основной тип данных – это скаляры, одно- и многомерные массивы, которые могут автоматически обрабатываться;
- основные операторы языка имеют ФОРТРАН-подобный вид.

4. ОСОБЕННОСТИ РЕАЛИЗАЦИИ СИСТЕМЫ PAW ДЛЯ MS DOS

4.1. Требования к операционной системе и аппаратуре

Система PAW и все ее компоненты в основном написаны на языке ФОРТРАН-77[12]. Для нормальной работы системы необходимы версия MS DOS не ниже 3.30 и следующее оборудование:

- i386/387 или i486 CPU;
- как минимум 6 Mb RAM;
- 4 Mb на "винчестере";
- любой графический адаптер (CGA, EGA, VGA), но желательно VGA;
- любое печатающее устройство или графопостроитель для получения "твердых" копий рисунков на бумаге.

Для компиляции были использованы трансляторы NDP-ФОРТРАН[13] и NDP-C фирмы MicroWay, Inc. Трансляторы этой фирмы позволяют создавать программы общим размером до 4 Гбайт и запускать их на счет в среде MS DOS с переключением процессора в "защищенный" режим с использованием 32-разрядных команд i386/486 CPU.

	Тип метафайла	Размер (Кбайт)	Время (с)
1	18 – Golden	183	11.2
2	RZ	33	1.5
3	-777 – LaTeX	146	43.7
4	-111 – *PostScript*	217	63.0
5	FZ (A-format)	24	2.8
6			1.4
7	18 – VGA		5.4

Таблица 1: Некоторые параметры метафайлов различного типа

4.2. Инициализация PAW

Когда система начинает работать, в первую очередь инициируется процедура запуска системы, которая показывает на дисплее текущую версию PAW

```
C:\ PAW
    ---
*******************************************
*                                         *
*                                         *
*     W E L C O M E   to   P A W          *
*                                         *
*                                         *
*    Version 1.14/02  13 March 1992       *
*                                         *
*                                         *
*******************************************
Workstation type (?=HELP) [CR]=18 :
```

и запрашивает тип графического монитора или *метафайла*. При этом пользователь получает подсказку о наиболее оптимальном режиме использования его графической платы. Оптимальный тип метафайла зависит от того, как предполагается в дальнейшем использовать результаты работы. В таблице 1 приведены некоторые характерные параметры различных типов метафайлов, полученные при запуске стандартного тестового примера для библиотеки HIGZ[14] на машине типа IBM PC AT 386/387/33Мгц. В таблице дано время формирования метафайла и его размер. Так как последовательные FZ-файлы не формируются непосредственно в ходе рисования и используются только для целей обмена информацией, в строках 5 и 6 таблицы дано время преобразования RZ-файла (поз.5) и ZEBRA-структуры, размещенной в оперативной памяти (поз.6), в FZ-exchange-формат с помощью команды `Toalpha` системы PAW. Для сопоставления приведено время вывода тех же рисунков непосредственно на экран с помощью команды `Picture\plot *`.

Примеры использования различных типов метафайлов приведены в следующих разделах.

Перед тем, как передать управление пользователю, система ищет файл с именем **pawlogon.kum**. Последний может содержать команды, которые необходимо выполнять каждый раз в начале работы системы: определение пользовательских файлов, "своих" команд, определение параметров для HPLOT и т.п. Например, в простейшем случае, этот файл может содержать следующий набор команд для вывода на экран текущего времени и даты запуска системы:

```
mess '****************************************'
mess '*                                      *'
mess '* Starting PAW session on '//$date//' at '//$time//' *'
mess '*                                      *'
mess '*                                      *'
mess '****************************************'
```

4.3. Использование клавиатуры и манипулятора "мышь"

Для повышения удобства ввода команд системы PAW в версии для MS DOS разработан специальный драйвер, который позволяет значительно облегчить работу с системой.

Драйвер позволяет для режимов 2 и 3 из нижеприведенного списка:

1. ввод команд;
2. ввод параметров для команд с автоматическим формированием команды;
3. ввод параметров для макрокоманд;
4. ввод координат графического указателя.

иметь 2 дополнительных стека. В результате в режиме ввода параметров после нажатия на клавиши "стрелка вверх"/"стрелка вниз" из стека будут извлекаться только значения, ранее введенные имено для соответствующих параметров. В режиме ввода команд при нажатии на те же клавиши будет извлекаться соответствующая строка из так называемого **history file** пакета KUIP. Напомним, что в этом файле постоянно хранится последовательность из N последних введенных "полных" команд, которые, таким образом, могут быть в любой момент просмотрены, скопированы, отредактированы и исполнены. Т.е., если исследователь вводил команду "в несколько приемов", используя подсказки системы меню, сокращенное название команды и значения параметров "по умолчанию", то из стека он извлечет всю команду целиком со всеми параметрами – без сокращений, и все параметры будут стоять на своих местах и иметь конкретные значения. Такой прием значительно облегчает работу с системой, имеющей большое число разнообразных команд с многочисленными параметрами.

Для перемещения графического указателя (графического курсора) можно использовать одновременно как алфавитно-цифровую клавиатуру, так и устройства типа "мышь" или "световое перо", либо только клавиатуру. Наличие в активном состоянии устройства типа "мышь" пакет определяет автоматически.

Для увеличения/уменьшения скорости (и изменения точности) движения графического курсора можно использовать клавиши [Ins]/[Del]. Последовательное использование клавиши [Ins] приводит к увеличению скорости перемещения курсора, а клавиши [Del] – к уменьшению ее и, как следствие, увеличению точности позиционирования.

Для перемещения курсора параллельно осям координат можно использовать клавиши "стрелки" клавиатуры, вместо левой кнопки "мыши" – клавишу [Esc], вместо правой – [Enter]. Дополнительно можно использовать клавиши [7] -- [Home], [9] -- [Pg Up], [1] -- [End], [3] -- [Pg Dw] для перемещения курсора в диагональных направлениях.

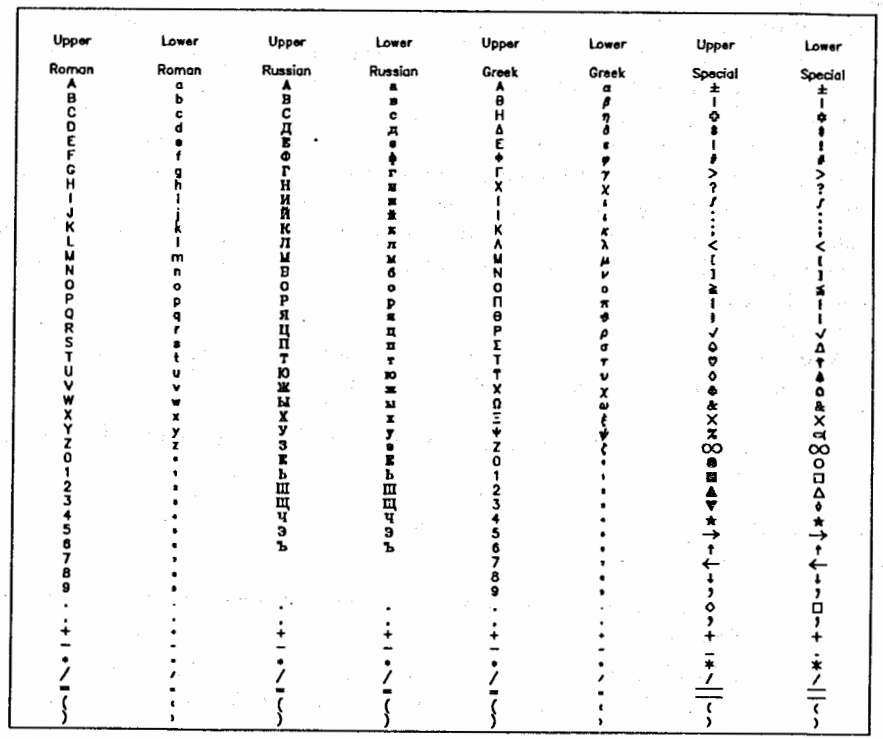

Рис. 2: Таблица транслитерации для русских и греческих букв и спец.символов

Возможности стандартного знакогенератора подсистемы HIGZ расширены. Для оформления рисунков пользователь может использовать русский алфавит. Предусмотрено два способа задания русских символов: в "альтернативной" кодировке IBM PC и с помощью управляющих символов и транслитерации. Во втором случае русские буквы кодируются соответствующими латинскими и берутся в двойные квадратные скобки (см. рис.2):

```
[[ ... текст на русском языке латинскими буквами ...]]
```

В этом случае прикладная программа или макрокоманда будут одинаково работать на любой ЭВМ независимо от принятого на той или иной машине или в операционной системе способа представления русского алфавита.

5. ПОДГОТОВКА РИСУНКОВ ДЛЯ ПУБЛИКАЦИИ С ПОМОЩЬЮ LaTeX на IBM PC

Включение рисунков в тексты публикаций, подготавливаемых с помощью системы LaTeX[15],– это область, в которой пользователи, как правило, сталкиваются с массой проблем[16]. Хотя последние версии пакета PAW включают возможность формирования метафайла непосредственно в формате системы подготовки публикаций LaTeX, реально воспользоваться этой возможностью при работе на IBM PC практически невозможно из-за чрезвычайной сложности метафайла: обычно версии LaTeX на PC просто не хватает оперативной памяти. При обработке метафайла в LaTeX-формате на машине VAX-8350 время генерации одной картинки может составить несколько часов работы. Понятно, что использование подготовленных пакетом PAW картинок в формате LaTeX и на машине VAX сильно ограниченно.

В данной работе предлагается другой путь[4], главная особенность которого заключается в том, что его можно практически использовать[5].

Общий алгоритм включает в себя (см.рис.3):

1. получение метафайла рисунка с помощью пакета PAW в GOLDEN-формате (PLT-формат)[6];
2. определение параметров размещения рисунка на странице, в том числе:
   (a) определение размера области для размещения рисунка на странице публикации;
   (b) определение необходимого коэффициента масштабирования и сдвига, (S_x, S_y, $Shift_x$, $Shift_y$) с помощью утилиты VIEW[7].
3. генерация OUT-файла в формате PCL (формат, используемый при выводе графической информации на лазерный принтер, например LaserJet II) с помощью программы PLOT[8];

[4] Авторы благодарят Федунова А.Г. за помощь в освоении и практической проверке предлагаемого алгоритма.

[5] Все иллюстрации в данном сборнике подготовлены описываемым ниже способом.

[6] Форматом 'Golden' будем называть здесь формат метафайлов, используемый в пакетах фирмы Golden Software Inc., например, SURFER и GRAPHER.

[7] VIEW - утилита, входящая в состав пакетов фирмы Golden Software Inc.

[8] PLOT - утилита, входящая в состав пакетов фирмы Golden Software Inc.

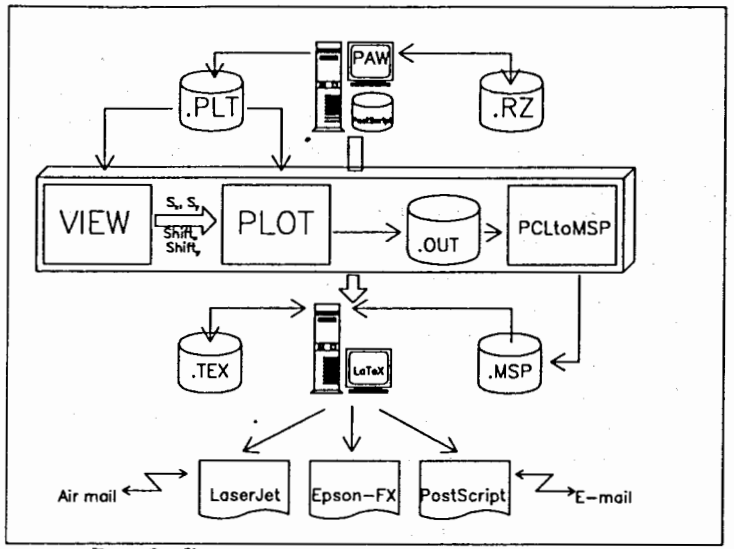

Рис. 3: Схема подготовки рисунков к публикации

4. преобразование OUT-файла из формата PCL в формат MSP с помощью программы PCLtoMSP;
5. Получение окончательного варианта статьи с помощью пакета LaTeX:
   (a) включение рисунка в MSP-формате в публикацию с помощью команд
   `\begin{figure}`, `\begin{picture}`,
   `\put(x,y){\special{em:graph filename.msp}}`
   `\end{picture}`, `\end{figure}`
   LaTeX и получение DVI-файла;
   (b) интерпретация DVI, получение готового к публикации оттиска на бумаге с помощью лазерного или обычного матричного принтера;
   (c) в случае, если необходима последующая передача по системе электронной почты, получение файла в формате PostScript.

Преимуществом данного алгоритма является его практическая реализуемость. Его использование **не требует** ресурсов суперЭВМ, и время, необходимое для реализации пунктов 2-5 на IBM PC AT 386/25, составляет всего несколько секунд.

Недостатком является невозможность передавать рисунки по системе электронной почты в организации, где нет пакета PAW. В этом случае рекомендуется комбинированный метод. На стадии подготовки текста необходимо воспользоваться предлагаемой здесь процедурой, а для пересылки окончательного варианта можно сгенерировать метафайл в формате LaTeX или PostScript.

5.1. Получение метафайла и "чернового" рисунка

Для того, чтобы получить метафайл рисунка, необходимо установить для PAW режим работы с метафайлом с помощью команды **METAFILE 10** (описание см. в руководстве по PAW). В результате работы пакета в режиме метафайла после завершения сеанса Вы в своей рабочей директории обнаружите ряд новых файлов с именем **META<n>.PLT**[9], где n – целое число. Каждый такой файл будет содержать один рисунок в формате, используемом в широко распространенной в ОИЯИ системе - GRAPHER/SURFER. Чтобы вывести содержимое этих файлов на графическое устройство (игольчатый принтер, лазерный принтер, графопостроитель и т.п.) без помощи пакета LaTeX, необходимо воспользоваться программой PLOT из вышеуказанного пакета. Такой способ удобен на ранних стадиях подготовки публикаций, когда окончательный внешний вид иллюстраций еще не сформировался и надо уметь быстро (без особых претензий на качество) получать копии рисунков на бумаге. С помощью утилиты PLOT можно не только получить "твердую" копию, но и преобразовать *метафайл* в формат систем AutoCAD или получить его на языке *PostScript*.

По команде

PLOT META<n>.PLT,

где n - порядковый номер рисунка, Вы получите "твердую" копию n-го рисунка на бумаге [10].

По команде

VIEW META<n>.PLT

можно еще раз просмотреть рисунок на экране монитора.

5.2. Определение параметров размещения рисунка

Для правильного и красивого размещения рисунка на странице Вашей публикации необходимого определить следующие параметры:

1. размер рисунка на странице публикации,
2. формат листа, используемый программой PLOT при генерации OUT-файла,
3. начальный размер рисунка, определяемый пакетом PAW.

Размер области, отводимой на странице под рисунок, определяется эстетическими соображениями. Для дальнейшей работы важно знать этот размер вдоль H_{pict} и поперек L_{pict} страницы в дюймах или сантиметрах.

Формат листа (H_{form} и L_{form}), используемый программой PLOT, можно определить и изменить, при необходимости, с помощью команды:

[9] В старой, версии 1.10, PAW для MSDOS метафайлы формировались с именем **PICT<n>.PLT**.

[10] Программа PLOT должна быть предварительно настроена на конкретный тип графического устройства. Предварительная настройка осуществляется по команде: **PLOT /i**.

PLOT /i.

С помощью команды

VIEW META<n>

можно просмотреть метафайл и определить начальные размеры рисунка в дюймах. Для того, чтобы определить размер, необходимо после завершения вывода рисунка на экран нажать клавишу **[ESC]** и выбрать позицию **Digitize** в меню. В результате на экране должен появиться графический курсор и его координаты в дюймах в левом нижнем углу. Подведя графический курсор к правому верхнему углу рисунка, можно узнать его координаты H_{plot} и L_{plot}.

Теперь можно вычислить значения коэффициентов масштабирования и сдвига (S_x, S_y, $Shift_x$, $Shift_y$)

$$\begin{aligned} S_x &= S_y = H_{pict}/H_{plot} \\ Shift_x &= 0 \\ Shift_y &= H_{form} - H_{plot} * S_y. \end{aligned} \tag{1}$$

Если для окончательного вывода предполагается использовать 9-игольчатый принтер типа Epson, то значения коэффициентов необходимо скорректировать:

$$\begin{aligned} S_x &= S_x/1.25 \\ S_y &= S_y/1.38. \end{aligned} \tag{2}$$

После всех вышеуказанных манипуляций Вы получите все необходимые исходные данные для правильного размещения Вашего рисунка.

5.3. Генерация OUT-файла

Получить OUT-файл в PCL-формате можно с помощью программы PLOT:

PLOT META<n>.PLT,

где n – порядковый номер рисунка. После запуска программы в диалоговом режиме необходимо ввести вычисленные заранее параметры.

Программа PLOT должна быть предварительно настроена на генерацию выходного файла в формате для лазерного принтера LaserJet (даже если Вы собираетесь печатать публикацию на печатающем устройстве типа Epson).

Предварительная настройка осуществляется по команде

PLOT /i.

После установки типа принтера необходимо выбрать следующие позиции в меню настройки PLOT:

```
2. Change Output port or disk file
   6. Send output to xxx.OUT where xxx is the name
      of the input file
```

5.4. Преобразование OUT-файла в MSP-формат

Преобразование OUT-файла в формат, необходимый для работы пакета LaTeX, осуществляется программой PCLtoMSP, входящей в состав утилит пакета LaTeX. Функции этой программы и порядок ее использования достаточно подробно описан в соответствующем DOC-файле. В нашем случае, как правило, достаточно набрать:

```
PCLtoMCP META<n>.OUT META<n>.msp
```

5.5. Включение рисунка в публикацию

После всех манипуляций мы, наконец, получаем файл, пригодный для подготовки публикации с помощью пакета LaTeX. Для того, чтобы поместить рисунок на отведенное ему место, необходимо в Ваш LaTeX-файл вставить следующую последовательность команд:

```
\begin{figure}[hc]
\unitlength 1in

\begin{picture}(<L_pict>,<H_pict>) \put(0,<H_pict>){\special{em:graph meta<n>.msp}}

\end{picture}
\caption{Схема подготовки рисунков к публикации}
\end{figure}
```

Далее работа с текстом строится в соответствии с инструкциями к пакету LaTeX[15].

Приведенный алгоритм может быть использован не только для рисунков, генерируемых пакетом PAW, но в других случаях. С п.2 его можно использовать для включения в публикацию рисунков, подготовленных с помощью пакетов фирмы GOLDEN (например, SURFER и GRAPHER), с п.4 - для включения рисунков, полученных с помощью любых пакетов, которые "умеют" выводить рисунки в промежуточный файл для лазерной печати.

6. ОРГАНИЗАЦИЯ ВЗАИМОДЕЙСТВИЯ РАБОЧЕЙ СТАНЦИИ НА БАЗЕ PC С ЦЕНТРАЛЬНОЙ ЭВМ

Локальная версия пакета PAW обеспечивает прием/передачу информации с ЭВМ (например VAX или ЕС ЭВМ) и ее последующую обработку. При этом передавать информацию с центральной машины на рабочую станцию можно с использованием **любых** доступных пользователю средств, начиная от прямого взаимодействия с использованием сетевых возможностей (Ethernet, Jinet[17], PCONLINE[18]) и кончая переносом файлов на дискетах.

Для обеспечения взаимодействия (интерактивного анализа и графической интерпретации данных) в рамках комплекса PC-Host можно рекомендовать следующую процедуру:

- Формирование метафайла в формате HIGZ на центральной ЭВМ;
- Преобразование RZ-файла в текстовый формат FZ (ASCII);
- Передачу файлов на PC;
- Интерактивный анализ и графическую интерпретацию (PAW PC);
- Получение "твердой" копии hard-copy на бумаге.

Если для формирования метафайла в формате HIGZ на центральной машине применяются программа моделирования GEANT3, пакеты HBOOK, HPLOT, которые для хранения *гистограмм*, графической информации (*рисунков*) и других объектов используют систему управления структурами данных ZEBRA и графический интерфейс HIGZ, то сформировать файл графической информации в RZ-формате можно, например, следующим образом, используя операторы языка ФОРТРАН:

в начале программы:

```
C =====>        Pictures stores to disk in RZ format
        LUNRZ=18
        CALL IGINIT(0)
        CALL IGZSET('Z')
        CALL IGSET('AURZ',1)
        CALL IZOPEN(LUNRZ,'PICTURE','TEST.RZ','AN',1024,ISTAT)
        CALL HPLINT(0)
```

– перед каждой картинкой:

```
C =====>        Make a new picture
        CALL IZPICT(' ','M')
```

– в конце программы:

```
C =====>        Write all pictures to disk
        CALL IZOUT('*',IIC)
```

где **LUNRZ** – логический номер устройства с файлом прямого доступа.

Для последующей передачи файла на ЭВМ другого типа (PC) полученный файл прямого доступа необходимо преобразовать в стандартный текстовый ASCII-
формат обмена данными пакета FZ системы ZEBRA. Сделать это можно

- средствами самой системы ZEBRA;
- с помощью команд пакета PAW;

- специальной программой преобразования RZ-файлов RTOA;
- с помощью вспомогательной процедуры.

Полученный в результате файл является "машинонезависимым" и может передаваться на другие типы ЭВМ (PC, VAX, ЕС ЭВМ и др.) любым доступным способом, включая "электронную почту".

Для удобства передачи графической информации с ЕС на PC (в рамках системы PAW) на языке KUIP в ЛВТА ОИЯИ разработаны макрокоманды GET и GETH[19], которые автоматически выполняют:

- передачу AFZ-файла с ЕС на PC;
- загрузку картинок (*гистограмм*) в память;
- выдачу на экран списка переданных картинок (*гистограмм*).

Необходимо отметить, что преобразование RZ-файлов в ASCII-формат необходимо только в двух случаях, если:

1. канал, по которому будет передаваться информация, не обеспечивает передачу двоичных файлов без искажений (например, E-mail);
2. хотя бы на одной из двух взаимодействующих ЭВМ пакет ZEBRA установлен в режиме **native** (см. п.3.4).

Если используемый Вами канал допускает надежную передачу двоичной информации и стандартный способ хранения информации на центральной ЭВМ для пакета ZEBRA – транспортный (exchange) [11], то необходимость преобразования RZ-файлов в ASCII-формат отпадает, и получать файлы на PC можно с помощью обычной процедуры копирования файлов (в обратную сторону это, как правило, неверно, т.е. нельзя просто копировать RZ-файлы с PC на VAX или ЕС ЭВМ).

7. ПРИМЕРЫ РЕШЕНИЯ ПРИКЛАДНЫХ ЗАДАЧ

7.1. Сравнение гистограмм с вычислением χ^2-функционала

Часто приходится сталкиваться с необходимостью сравнения двух гистограмм: например, гистограммы, содержащей экспериментальные данные, и гистограммы, содержащей значения функции, используемой для описания этих данных. В рамках PAW существует возможность сравнения двух гистограмм командой:

[11] Стандартная версия пакета ZEBRA для PC генерируется с режимом хранения – транспортный (exchange)


```
PAW > HI/OP/DIFF [id1] [id2]
```

Однако эта операция не дает значения функционала χ^2, в то время как в процедурах, выполняющих фитирование, степень совпадения функции с экспериментальным распределением определяется именно значением χ^2. Поэтому полезно иметь макрокоманду, позволяющую вычислить этот функционал.

Математически расчетная формула имеет известный вид:

$$\chi^2 = \sum_{i=1}^{n} \frac{(C(i) - F(i))^2}{E(i)^2},$$

где суммирование проводится по числу каналов гистограммы,$C(i)$ – содержимое i-го канала, $E(i)$ – ошибка $C(i)$, $F(i)$ – значение функции в этом канале.

Распространенным критерием является значение χ^2 на степень свободы. Количество степеней свободы распределения определяется числом каналов гистограммы и, если функция параметрическая, числом свободных параметров. В представленной макрокоманде предусмотрена возможность задания любого их количества.

```
************************************************************************
* Calculates chisqare between two histograms and chi2/d.f. for some
* free parameters. Requires two positional parameters: identifier
* of the basic histogram and identifier of the "function"-histogram.
* So the call to this macro must be of the following art:
*  PAW > ...
*  PAW > exec chisq [1] [2]
*  PAW > ...
* Here ID=[1] have the basic histogram and ID=[2] have the histogram
* which contains the comparing function.
************************************************************************
macro chisq
    message ' Type the number of free parameters (NFP): '
    read NFP                        | reading the number of free parameters
    nf=[nfp]+1                      | calculating new constant (see below)
    ve/cr cont1(50)                 | creating vectors to
    ve/cr cont2(50)                 | fill them with contents
    ve/cr erro1(50)                 | of histograms
    hi/get/cont [1] cont1           | filling CONT1 with contents of histogram[1]
    hi/get/erro [1] erro1           | filling ERRO1 with errors of histogram[1]
    hi/get/cont [2] cont2           | filling CONT2 with contents of histogram[2]
    vsub cont1 cont2 ressub         | creating vector RESSUB=CONT1-CONT2
    vdiv ressub erro1 resdiv        | creating vector RESDIV=RESSUB/ERRO1
    vmult resdiv resdiv res         | creating vector RES=RESDIV*RESDIV
    sigma chi=sum(res)              | summing elements of RES to obtain chisqare
    i=0                             | counter
loop:                               | loop over histogram channels
    i=[i]+1                         | counter
    if cont1([i]) <> 0 goto loop    | checking if channel [i] contains zero
```

```
    np=[i]-[nf]                 | number of degrees of free NP=I-nf, where
*                               | nf is number of free parameters plus 1
    sigma cdf=chi/[np]          | calculating chisqare/degree of free
    c=chi(50)                   | assuming CHI(50) to the variable C
    cd=cdf(50)                  | assuming CDF(50) to the variable CD
    mess ' Difference between histograms ' [1] ' and ' [2] ' :'
    mess ' Chisqare=' [c]       | printing a message
    mess ' Chi2/df =' [cd]      | printing a message
    ve/de *                     | deleting all vectors
return
*******************************************************************************
```

Запуск макрокоманды осуществляется командой

```
PAW > EXEC CHISQ id1 id2,
```

где **CHISQ** – имя файла, содержащего макрокоманду, **id1** – номер гистограммы, содержащей "базовое" распределение, и **id2** – номер сравниваемой гистограммы.

Обе эти гистограммы должны существовать в текущей директории . Вычисление χ^2 осуществляется посредством выполнения арифметических операций над *векторами*, поэтому результат любой операции есть *вектор*, а не переменная. В представленной программе для подобных операций используется не только пакет VECTOR, но и пакет SIGMA. В частности, команда **SIGMA CHI=SUM(RES)** создает новый *вектор* **CHI**, такой же размерности, что и *вектор* **RES**, но содержащий бегущую сумму элементов **RES**, т.е. интересующая нас сумма **всех** элементов содержится в последнем элементе **CHI**. **Операция SIGMA CDF=CHI/[NP]** создает *вектор* **CDF**. Каждый его элемент есть результат деления соответствующего элемента **CHI** на переменную **NP**. Для дальнейшей работы удобно ввести две новые переменные **C** и **CD**, присваивая им, соответственно, значения последних элементов **CHI** и **CDF** . Это позволяет вывести результат на экран посредством команды **MESSAGE**, которая оперирует лишь с переменными. Такие операции с *векторами* очень удобны в том случае, когда размер гистограммы неизвестен: при выполнении операции деления *вектора* на *вектор* результатом деления на ноль является ноль, что избавляет от многих проблем.

Эти замечания о работе с *векторами* представляют собой общую проблему, с которой нередко приходится сталкиваться, пользуясь PAW.

7.2. Сложение гистограмм с учетом ошибок

На практике часто бывает необходимо просуммировать несколько по-разному отнормированных *гистограмм*. Суммирование *гистограмм* предусмотрено в рамках PAW: оно выполняется командой

```
PAW > ADD id1 id2 id3 w1 w2.
```

Однако эта команда дает правильный результат, если известен вес каждой гистограммы и не надо учитывать ошибки. Для точного суммирования необходимо иметь процедуру, вычисляющую вес и пересчитывающую ошибки. Здесь

представлена программа, проводящая суммирование двух *гистограмм* с учетом ошибок.

```
*******************************************************************************
*      This macro sums up two normalized histograms
*   Requires existance of at least one histogram in current directory
*******************************************************************************
macro sumhis
  ve/de *                               | cleaning memory
  message ' Enter the first histogram ID :'  | printing message
  read id1                              | prompting for id1
  message ' Enter the second histogram ID :' | printing message
  read id2                              | prompting for id2
  ve/cr cont1(100)                      |----------------------------
  ve/cr erro1(100)                      | creating some
  ve/cr cont2(100)                      | necessary vectors
  ve/cr erro2(100)                      |----------------------------
  hi/get/cont [id1] cont1               | filling CONT1 with contents of id1
  hi/get/erro [id1] erro1               | filling ERRO1 with errors of id1
  hi/get/cont [id2] cont2               | filling CONT2 with contents of id2
  hi/get/erro [id2] erro2               | filling ERRO2 with errors of id2
*==============================================================================
*      some mathematics
*==============================================================================
  vmult erro1 erro1 errI                | it is 1/N1
  vmult erro2 erro2 errII               | it is 1/N2
  vmult erro1 erro2 errmul              | it is sqrt(1/(N1*N2))
  vadd errI errII errsum                | it is (1/N1)+(1/N2)
  vdiv errII errsum weig1               | weight for first histogram
  vdiv errI errsum weig2                | the same for second one
  sigma errsq=sqrt(errsum)              | it is sqrt(1/(N1+N2))
  vdiv errmul errsq erro3               | creating ERRO3 with new errors
  vmult cont1 weig1 sum1                | weighted histogram-1
  vmult cont2 weig2 sum2                | weighted histogram-2
  vadd sum1 sum2 cont3                  | creating CONT3 with resulting
*                                       | histogram contents
*==============================================================================
  hi/copy [id1] 100 'Result of SUMHIS'  | creating histogram 100, the
*                                       | same as ID1 but with new title
  hi/put/cont 100 cont3                 | filling id=100 with vector CONT3
  hi/put/erro 100 erro3                 | filling id=100 with vector ERRO3
  ve/de *                               | cleaning memory
  hi/file 3 result.plot ! n             | opening new file RESULT.PLOT
  hrout 0                               | writing all histograms on
*                                       | opened file
  close 3                               | closing opened file
return
*******************************************************************************
```

Программа запускается командой

```
PAW > EXEC SUMHIS
```

Для ее работы требуется наличие в текущей директории хотя бы одной гистограммы. Процедура запрашивает номера суммируемых распределений в процессе выполнения команды **READ**. Из каждой гистограммы извлекаются ее содержимое и ошибки. Содержимое затем складывается по каналам с учетом рассчитанного веса (предполагается, что имеем дело с неким распределением, ошибки которого имеют лишь статистический характер). Новые ошибки вычисляются из ненормированного нового содержимого.

Все операции с векторами, в которых записано содержимое и ошибки гистограмм, проводятся посредством команд PAW из пакета VECTOR. Каждая такая операция создает новый *вектор* (если он не был создан заранее). Вектора, содержащие итоговое распределение и ошибки, записываются во вновь создаваемую гистограмму с номером 100. Так как пользователь может ничего не знать о характере исходных гистограмм, то не всегда просто организовать результирующую гистограмму. Здесь предложен простой способ решения этой проблемы: учитывая то, что команда копирования гистограмм

```
PAW > HI/COPY id1 id2 newname
```

создает копию существующей гистограммы с другим номером и именем, организуется вспомогательная гистограмма с такими же параметрами, что и у исходной. В нее записывается затем результат, а в конце работы все распределения запоминаются во вновь открываемом внешнем файле прямого доступа **RESULT.POT**.

В качестве примера использования приведенных выше макрокоманд можно рассмотреть процедуру, создающую гистограммы и проводящую с ними операции сложения и вычисления χ^2 :

```
* Example of application macros SUMHIS and CHISQ
macro prepare
 1d 10 'Hist1' 20 0. 20.    | creating histogram 10
 1d 20 'Hist2' 20 0. 20.    | creating histogram 20
 ve/cr cont1(20)            |--------------------------------
 ve/cr erro1(20)            | creating some
 ve/cr cont2(20)            | necessary vectors
 ve/cr erro2(20)            |--------------------------------
*
* NOTE that SIGMA can't be inserted into MACRO like COMIS !
*
 sigma cont1=rndm(erro1)    | filling CONT1 with random numbers
 sigma erro1=1./sqrt(cont1)| filling ERRO1 with statistical errors
 sigma cont1=cont1*10.      | "renormalizing" of CONT1
 sigma cont2=rndm(cont1)    | filling CONT2 with random numbers
 sigma erro2=1./sqrt(cont2)| filling ERRO2 with statistical errors
 sigma cont2=cont2*20.      | "renormalizing" of CONT2
 hi/put/cont 10 cont1       | filling histogram 10 with contents
 hi/put/erro 10 erro1       | filling histogram 10 with errors
```



```
 hi/put/cont 20 cont2       | filling histogram 20 with contents
 hi/put/erro 20 erro2       | filling histogram 20 with errors
 hi/li                      | checking the directory
 EXEC sumhis                | executing MACRO SUMHIS to summ up 10 and 20
 add 10 20 150              | adding 10 and 20 without weights
 hi/copy 150 200 'Result of HI/OP/ADD' | copying 150 to 200 to rename
 hi/de 150                  | deleting id=150
 hi/li                      | checking the directory
 EXEC chisq 10 20           | executing MACRO CHISQ to calculate
*                           | chisqare between 10 and 20
 EXEC chisq 100 200         | doing the same with 100 and 200
 zone 2 2                   | dividing screen into 4 parts
 hi/plot 10 surf            |-----------------------------------------
 hi/plot 20 surf            | plotting histograms connecting
 hi/plot 100 surf           | channel contents by a line
 hi/plot 200 surf           |-----------------------------------------
 return
```

Эта процедура представляет собой элементарный способ заполнения гистограмм содержимым неких векторов. В данном случае *вектора* **CONT1** и **CONT2** заполняются случайными числами: такая возможность предусмотрена в рамках пакета SIGMA. Надо отметить, что хотя в интерактивном режиме работы с PAW можно инициализировать этот пакет с помощью команды

```
PAW > APPLication SIGMA,
```

но в макрокоманде этот способ не действует (в отличие от случая работы с пакетом COMIS). Поэтому в KUMAC-файле перед каждой строкой, в которой предполагается обращение к SIGMA, надо писать командное слово "**sigma**", тогда как при интерактивном общении можно это не делать.

Для имитации разной нормировки содержимое векторов **CONT1** и **CONT2** умножается на разные коэффициенты, а ошибки вычисляются из **начального** содержимого (они находятся в векторах **ERRO1** и **ERRO2**).

После суммирования гистограмм 10 и 20 процедурой **SUMHIS** для сравнения производится сложение этих же гистограмм стандартной командой PAW, правда, без учета веса каждой из них, который, впрочем, можно и не знать заранее. Кроме неудобств с вычислением веса, команда **HI/OP/ADD** имеет еще один недостаток: она создает результирующую гистограмму с тем же именем, что и первая гистограмма из списка параметров команды, а это не всегда желательно. Поэтому приходится эту результирующую гистограмму переименовывать тем же способом, который обсуждался выше.

После выполнения всех операций в текущей директории находятся четыре гистограммы: две исходные (**HIST1** и **HIST2**) и два результата суммирования разными способами. Для одновременного просмотра всех этих гистограмм экран делится на четыре зоны командой **ZONE**. При работе команды **HI/PLOT** вывод гистограмм на экран осуществляется в следующей последовательности: первая изображается в верхней левой зоне, вторая – в верхней правой, третья – в ниж-

ней левой и четвертая – в нижней правой. Для возврата экрана в режим одной зоны следует набрать команду **ZONE** без параметров.

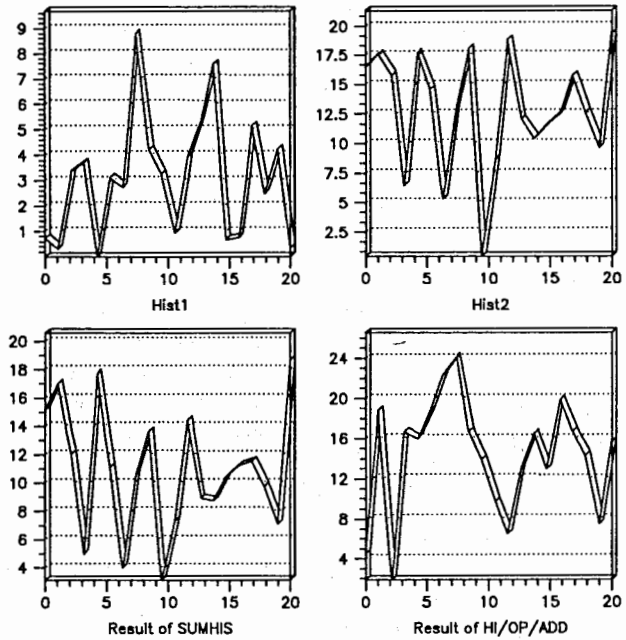

Рис. 4: Пример работы описанных выше процедур.

7.3. Построение диаграмм Фейнмана

В данном разделе демонстрируется возможность изображения средствами PAW, с помощью пакета GRAPHICS, сложных рисунков.

7.3.1. Построение фотонного пропагатора

Поставлена задача изображения средствами PAW фотонного пропагатора для диаграмм Фейнмана. Для этого используется существующая в рамках пакета GRAPHICS возможность изображения дуг окружностей :

```
PAW > ARC x y r1 r2 phi1 phi2
```


Представляемая макрокоманда требует задания шести параметров:

```
PAW > EXEC WAVE [1] [2] [3] [4] [5] [6]
```

где **WAVE** – имя **KUMAC**-файла, содержащего макрокоманду, и следующие параметры:

[1] – длина горизонтальной проекции пропагатора либо, если линия вертикальна, ее длина (в т.н. WC-координатах);

[2] – x-координата начала линии;

[3] – y-координата начала линии;

[4] – угол между горизонталью и направлением распространения волны (в градусах);

[5] – 1, если из вершины выходит одна линия и 2, если "вилка";

[6] – количество периодов волны на единицу длины.

```
*************************************************************************
*  Draws photon propagators in any direction and any length,
* requires six positional parameters:
*  PAW > ...
*  PAW > EXEC WAVE [1] [2] [3] [4] [5] [6]
*  PAW > ...
*  Here parameters are:
*    [1] - horizontal projection length or, if line is vertical, its length
*    [2] - abscissa of the beginning point (in World Coordinates)
*    [3] - ordinate of this point (in World Coordinates)
*    [4] - angle between horizont and direction of the wave (in degrees)
*    [5] - 1 if single propagator and 2 if branching
*    [6] - number of periods of wave per length unit
*************************************************************************
macro wave
  length=[1]                          |----------------------------------
  X=[2]                               | assuming parameters values
  Y=[3]                               | to variables
  angle=[4]                           |----------------------------------
begin:                                | marked position
  flag=0                              | flag controlling either 1 or 2 line
  grad=$sigma(acos(-1.)/180.)         | recounting degree<->radian
  r1=[grad]*[angle]                   | angle in radians
  if [angle]<>90 goto 10              | check whether line is vertical
  length1=[length]                    | if vertical,then length is absolute
  goto 20                             | going to mark 20:
10:                                   | marked position
  length1=$sigma([length]/cos([r1]))  | calculating absolute length of line
20:                                   | marked position
```

```
  n1=[length1]*[6]                    | calculating boundary for drawing
  ve/cr n3(1) i [n1]                  | making this_
  num=n3(1)                           | _boundary integer
  radius=$sigma([length1]/4/[num])    | calculating radius of half-circles
  delta=[radius]+[radius]             | calculating distance between centre
branch:                               | marked position
  rang=[grad]*[angle]                 | recalculating angle in radians
*                                     | (necessary for two lines case)
  i=0                                 | counter
  pang=[angle]+180                    |-----------------------------------
  mang=[angle]-180                    |
  a1=[angle]                          | angles for drawing
  a2=[pang]                           | an arc of circle
  a3=[mang]                           |
  a4=[angle]                          |-----------------------------------
  cang=$sigma(cos([rang]))            | cosinus of angle
  sang=$sigma(sin([rang]))            | sinus of angle
  posX=$sigma([X]+[radius]*([cang]))  | abscissa of the centre of arc
  posY=$sigma([Y]+[radius]*([sang]))  | ordinate of this centre
  xe=[X]+[length]                     | abscissa of the end of line point
  ye=$sigma([Y]+([sang])*[length]/[cang])|ordinate of this point
  dc=[delta]*[cang]                   | horizontal projection of DELTA
  ds=[delta]*[sang]                   | vertical projection of DELTA
loop:                                 | loop
  j=[i]+[i]                           | counter for "upper" arc
  k=[j]+1                             | counter for "lower" arc
  a=[dc]*[j]                          | x-shift for "upper" arc centre
  b=[ds]*[j]                          | y-shift for "upper" arc centre
  c=[dc]*[k]                          | x-shift for "lower" arc centre
  d=[ds]*[k]                          | y-shift for "lower" arc centre
  xn=[posX]+[a]                       | abscissa for "upper" arc centre
  yn=[posY]+[b]                       | ordinate for "upper" arc centre
  xc=[posX]+[c]                       | abscissa for "lower" arc centre
  yc=[posY]+[d]                       | ordinate for "lower" arc centre
  if [flag]=1 goto 50                 | checking flag
  goto 60                             | going to mark 60:
50:                                   | marked position
  a1=[mang]                           |-----------------------------------
  a2=[angle]                          | reversing angles
  a3=[angle]                          | for second branch
  a4=[pang]                           |-----------------------------------
60:                                   | marked position
  arc [xn] [yn] [radius] ! [a1] [a2]  | drawing "upper" arc
  arc [xc] [yc] [radius] ! [a3] [a4]  | drawing "lower" arc
  i=[i]+1                             | counter
  if [i]<[num] goto loop              | check whether counter reachs boundary
  if [flag]=1 goto endl               | checking flag
  if [5]=1 goto endl                  | checking needed number of lines
  angle=0-[angle]                     | reversing angle for second branch
```



```
  flag=1                              | changing flag
  goto branch                         | going to mark branch:
endl:                                 | marked position
  ve/de coor                          | cleaning memory
  ve/cr coor(4) r [X] [Y] [xe] [ye]   | creating vector COOR with
*                                     | coordinates of branch
  ve/de n3                            | cleaning memory
return
*******************************************************************************
```

Следует обратить внимание на работу с пакетом SIGMA в приведенном примере. При используемой здесь записи операторов пакет оперирует не с *векторами*, а с переменными. Его использование необходимо, т.к. функции `cos(x)`, `sin(x)`, а также арифметические операции, содержащие более одного действия, выполняются лишь в рамках SIGMA.

Отметим тот факт, что PAW не различает тип переменных по их названию, а описание типа переменных не осуществляется непосредственно. Поэтому в данной программе операция выделения целой части выполняется посредством создания *вектора*, элементы которого уже могут различаться по типу.

Используя возможности PAW, можно изображать и другие пропагаторы, в том числе и глюонные. Для построения последних в представленную процедуру достаточно ввести некоторые корректировки в углы раствора и радиусы дуг.

7.3.2. Построение диаграммы Фейнмана

Представленный выше файл `WAVE.KUM` позволяет изобразить один фотонный пропагатор (или симметричную относительно горизонтали пару) и его можно легко использовать для изображения реальных диаграмм. В этой связи возникает задача создания макрокоманды, способной облегчить работу по построению диаграмм. Здесь приводится вариант такой макрокоманды, не требующий задания дополнительных параметров, хотя при необходимости можно написать более универсальную программу.

```
*******************************************************************************
*  Draws a Feynman diagram using WAVE.KUMAC to draw photon propagators
*******************************************************************************
macro pict
opt zfl                           | setting pictures to put in Z data base
set xsiz 14                       | setting horizontal size (World Coordinates)
set ysiz 4                        | setting vertical size (World Coordinates)
clr                               | cleaning screen
next                              | creating a new picture
splci 3                           | set line colour index
exec wave 1 3.5 1.5 60 1 4.5      | drawing a propagator
exec wave 1 9.5 1 60 1 4.5        | drawing another propagator
splci 7                           | set other line colour index
line 4.5 1.5 6. 2.                | drawing a line
line 12.5 1.5 14. 2.              | drawing a line
```

```
arrow 3.9 4.5 1.3 1.5 0.2       | drawing an arrow
arrow 11.9 12.5 1.3 1.5 0.2     | drawing an arrow
splci 5                         | set a new line colour index
arc 3.5 1. 0.5 ! 0 360          | drawing a circle
arc 11.5 1. 0.5 ! 0 360         | drawing another circle
splci 7                         | reset line colour index
arrow 0. 1.5 1. 1. 0.2          | drawing an arrow
line 1.5 1. 3. 1.               | drawing a line
arrow 4. 5. 1. 1. 0.2           | drawing an arrow
line 5. 1. 6. 1.                | drawing a line
arrow 8. 9. 1. 1. 0.2           | drawing an arrow
arrow 9. 10. 1. 1. 0.2          | drawing an arrow
line 10. 1. 11. 1.              | drawing a line
arrow 12. 13. 1. 1. 0.2         | drawing an arrow
line 13. 1. 14. 1.              | drawing a line
arrow 3.9 4.5 0.7 0.5 0.2       | drawing an arrow
arrow 11.9 12.5 0.7 0.5 0.2     | drawing an arrow
line 4.5 0.5 6. 0.              | drawing a line
line 12.5 0.5 14. 0.            | drawing a line
text 0.4 0.6 'p' 0.35           |-----------------------------------
text 8.4 0.6 'p' 0.35           |
text 5.5 2.2 'p?<1>' 0.3        |
text 5.5 0.45 'p?<n>' 0.3       | typing a text in
text 13.5 2.2 'p?<1>' 0.3       | certain points
text 13.5 0.45 'p?<n>' 0.3      |
text 4.6 2.8 '[k]' 0.3          |
text 10.6 2.1 '[k]' 0.3         |
text 6.8 0.8 '=' 0.5            |-----------------------------------
return
*****************************************************************************
```

Вызов программы осуществляется командой

PAW > EXEC PICT,

где PICT – имя KUMAC-файла с этой процедурой. Получающаяся диаграмма сохраняется в виде *рисунка* в памяти и может быть записана как в *PICTURE*-файл, так и в любой *метафайл*. Для создания *рисунка* здесь не используется команда PAW> PICT/CREATE, а применяется команда NEXT, которая удобна тем, что создает новый рисунок с параметрами по умолчанию, вне зависимости от того, создавались ли рисунки до этого.

Т.к. координаты текстовых блоков трудно указать заранее, предлагается определять их с помощью команды VLOC, позволяющей найти координаты нужной точки по предварительно нарисованному изображению. Именно так были получены представленные здесь цифры.

В заключение стоит еще раз обратить внимание на необходимость использовать команду создания *рисунка* Picture/Create ... перед началом его формирования. Этим работа с *рисунками* отличается от графического вывода специальных объектов – *гистограмм* с помощью команды Hi/Plot ... , т.к. после

команды OPT ZFL для каждой гистограммы, изображаемой командой Hi/Plot, автоматически создается "свой" *рисунок*.

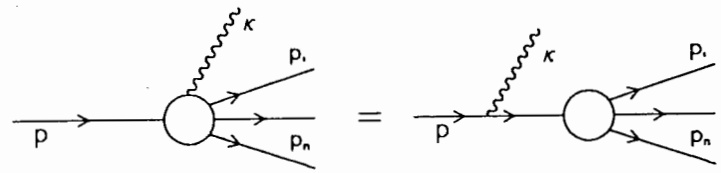

Рис. 5: Пример диаграммы Фейнмана

7.4. Вычисление и графическое представление волновых функций атома водорода с гармоническим потенциалом на трехмерной сфере

При решении различных физических и математических задач часто встречаются с проблемой быстрого графического представления результатов вычислений некоторых функций нескольких переменных. Система PAW позволяет сделать это без дополнительных усилий и получить графики в удобном для редактирования виде.

Описанный ниже пример взят из работы [20], вычислительная и графическая часть которой выполнена одним из авторов.

Рассмотрим шредингеровские функции атома водорода с гармоническим потенциалом – решением уравнения Лапласа на трехмерной сфере радиуса R, которые в пределе больших R соответствуют кулоновским функциям обычного атома водорода в "плоском" трехмерном пространстве. Для этого необходимо вычислять и строить графики для функций

$$\Psi_{nl\sigma}(\chi, R) = D_{nl\sigma}(R)(2\sigma \sin\chi)^l \exp(-\imath\chi(n-l-1-\imath\sigma))\times$$

$$\times {}_2F_1(-n+l+1, l+1+\imath\sigma, 2l+2, 1-e^{2\imath\chi}).$$

Здесь

$$D_{nl\sigma}(R) = \{(2\pi)^{-1/2}|\Gamma(l+1-\imath\sigma)|e^{\pi\sigma/2}\sigma^{1/2-(l+1)}\}\times$$

$$\times 2\sqrt{\frac{\sigma}{n}\frac{n^2+\sigma^2}{R^3}}\sqrt{\frac{(n+l)!}{(n-l-1)!}}\frac{1}{(2l+1)!}$$

$$\sigma = \frac{R}{an}, a = \frac{\hbar^2}{\mu\alpha}, \alpha = e^2 Z(Z=1),$$

а ${}_2F_1(-n+l+1, l+1+i\sigma, 2l+2, 1-e^{2ix})$ – гипергеометрическая функция. Соответствующие функции при различных значениях n, R и $r = \chi R, \chi \in [0, \pi]$, позволяющие проследить предельный переход, вычислялись в рамках системы PAW.

Для этого удобно воспользоваться командой PAW >fun2. Ниже приводится текст подпрограммы функции, вычисляющей нужные величины.

```
      function psi(x,y)
c*****************************************
c                                        *
c     calculate wave functions for n,l   *
c                                        *
c*****************************************
      vector ipar(2)
      PI = 3.1415926
      rb=y
      n=ipar(1)
      l=ipar(2)
      sig=rb/(1.*n)
      r=(x-0.0785)*rb
      lm1=-l-1
      l2=2*l
      d=1./sig**l*p(l,sig)/sqrt(1.-exp(-2.*pi*sig))*2.*
     * sqrt(sig/(1.*n)*(n*n+sig*sig)/(rb*rb*rb))*sqrt(fac(n,l)
     *)/fac1(l2,1)
      if (r.eq.0..and.l.eq.0) then
         dd = 1.
         ddo=1.
      else
         dd=(2.*sig*sin(r/rb))**(l)
      end if
       psi=d*dd*exp(-sig*r/rb)*sf(n,l,sig,r,rb)
       return
       end
      function fac(n,l)
          .
          .
          .
       return
       end
      function fac1(n,l)
          .
          .
          .
       return
       end
      function p(l,sig)
          .
          .
```



```
       return
       end
      function sf(n,l,sig,r,rb)
      complex ez,zz,b2,f21,a,a1,sss,c1,c2,c3
       nl1=-n+l+1
       b1=real(nl1)
       l1=l+1
       bl=real(l1)
       b2=cmplx(bl,sig)
       b3=2.*l+2.
       zz=(1.,0)-cexp(2.*(0.,1.)*r/rb)
       f21=(1.,0.)
       s=1.
       a=(1.,0.)
 1    b1s=b1+s-1.
      c1=cmplx (b1s,0.)
      ss=s-1.
      sss=cmplx(ss,0.)
      c2=b2+sss
      b3s=(b3+s-1.)*s
      c3=cmplx(b3s,0.)
      bd=abs(b1s)
      cd=bd-0.5
      if(cd) 6,5,5
 5    a1=a*c1*c2*zz/c3
      f21=f21+a1
      a=a1
      s=s+1.
      go to 1
 6    f21=f21
      ez=cexp((0.,1.)*b1*r/rb)
      f21=f21*ez
      sf=real(f21)
      return
      end
```

Вычисления проводятся в системе PAW для различных значений параметров n, l. Так как команда `fun2` воспринимает только параметры x, y, то остальные параметры задаются через вектор `vec/cr ipar(2) i` и новые значения присваиваются в процессе работы командой `vec/inp ipar n l`, где `n` и `l` – новые значения параметров.

Результаты вычислений получаются в виде двумерных "картинок", которые хранятся в памяти под номерами гистограмм и именами "картинок". Для компоновки рисунков, размещения надписей и т. д. используются возможности PAW.

Ниже приведен пример работы и представлены результаты работы приведенной последовательности команд.

```
vec/cr ipar(2) i                    ! создать вектор
```

```
vec/inp ipar 4 0                    ! присвоить значения элементам вектора
fun2 40 psi.for 20 0 3.14 20 1 20   ! нарисовать функцию в 20 точках
                                    ! по x ( 0<x<3.14 ) и в 20 точках
                                    ! по y ( 1<y<20)
vec/inp ipar 4 1                    ! присвоить новые значения  n,1
fun2 41 psi.for 20 0 3.14 20 1 20           .
vec/inp ipar 4 2                    ! присвоить новые значения  n,1
fun2 42 psi.for 20 0 3.14 20 1 20           .
vec/inp ipar 4 3                    ! присвоить новые значения  n,1
fun2 43 psi.for 20 0 3.14 20 1 20           .
opt utit                            ! обеспечить возможность подписать
                                    ! рисунок
opt zfl                             ! обеспечить возможность
                                    ! рисования "картинок"
hi/pl 40 surf                       ! создать "картинку" pict1
                                    ! из гистограммы 40
hi/pl 41 surf                               .
hi/pl 42 surf                               .
hi/pl 43 surf                               .
pi/li                             ! распечатать список "картинок"
pi/cr sum                         ! создать "картинку" с названием
                                  ! sum
pi/mer pict1 0 0.35 0.5           ! поместить в "картинку" sum
                                  ! pict1, уменьшенную в 2 раза, в
                                  ! левый верхний угол
pi/mer pict2 0.35 0.35 0.5        ! поместить в "картинку" sum
                                  ! pict2, уменьшенную в 2 раза, в
                                  ! правый верхний угол
pi/mer pict3 0 0 0.5              ! поместить в "картинку" sum
                                  ! pict3, уменьшенную в 2 раза, в
                                  ! левый нижний угол
pi/mer pict4 0.35 0 0.5           ! поместить в "картинку" sum
                                  ! pict4, уменьшенную в 2 раза, в
                                  ! правый нижний угол
vloc x y                          ! определить координаты первой надпи-
                                  ! си
text x(1) y(1) '[Y]?40' 0.3       ! поместить надпись в отмеченную
                                  ! координату
text x(2) y(2) '[Y]?41' 0.3
text x(3) y(3) '[Y]?42' 0.3
text x(4) y(4) '[Y]?43' 0.3
vloc x y
text x(3) y(3) 'R' 0.3
text x(2) y(2) 'R' 0.3
text x(1) y(1) 'R' 0.3
vloc x y
text x(1) y(1) 'R' 0.3
vloc x y
text x(1) y(1) '[h]' 0.3
```



```
text x(2) y(2) '[h]' 0.3
vloc x y
text x(1) y(1) '[h]' 0.3
meta 10                           ! открыть файл на запись "картинки"
pi/pl sum                         ! записать "картинку" sum в виде .plt
                                  ! файла
pi/fi 1 nnew.rz ! n               ! открыть rz-файл для сохранения
                                  ! результатов  в виде банка "картинок"
izout *                           ! записать все "картинки" в файл
```

Ниже приведен результат вычислений.

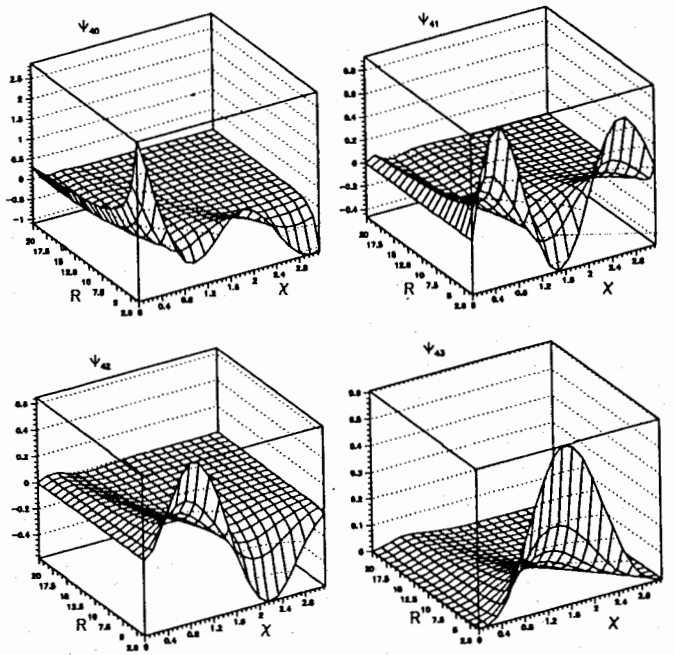

Рис. 6: Волновые функции атома водорода

7.5. Использование PAW для графического представления двумерной информации

Пользователь, работающий в системе PAW, может легко использовать ее для рисования графиков и картинок по данным, записанным на внешнем файле в форматном виде. (Например, если ему удобно, вместо пакета GRAPHER(SURFER))

Приведем здесь пример такого использования PAW.

Пусть необходимо построить двумерную поверхность $Z(x,y)$, посчитанную на равномерной сетке по x в 101 точке и в 9 - по y ($xmin = 0$, $xmax = 150$, $ymin = 0.01$, $ymax = 0.09$). Данные записаны в последовательный файл **f:\pol\new\fort.11** по формату **(21x,e16.9)** в следующем порядке: $Z(1,1),\ldots,$ $Z(101,1)$, $Z(1,2),\ldots,$ $Z(101,2),\ldots,$ $Z(1,9),\ldots,Z(101,9)$. Для построения поверхности используем команды PAW:

```
ve/read z f:\pol\new\fort.11 21x,e16.9
2d 10 'biexiton' 101 0 150 9 0.01 0.09
hi/put/cont 10 z
hi/pl 10 surf
opt nbox
meta 10
hi/pl 10(1:30,1:9) surf
```

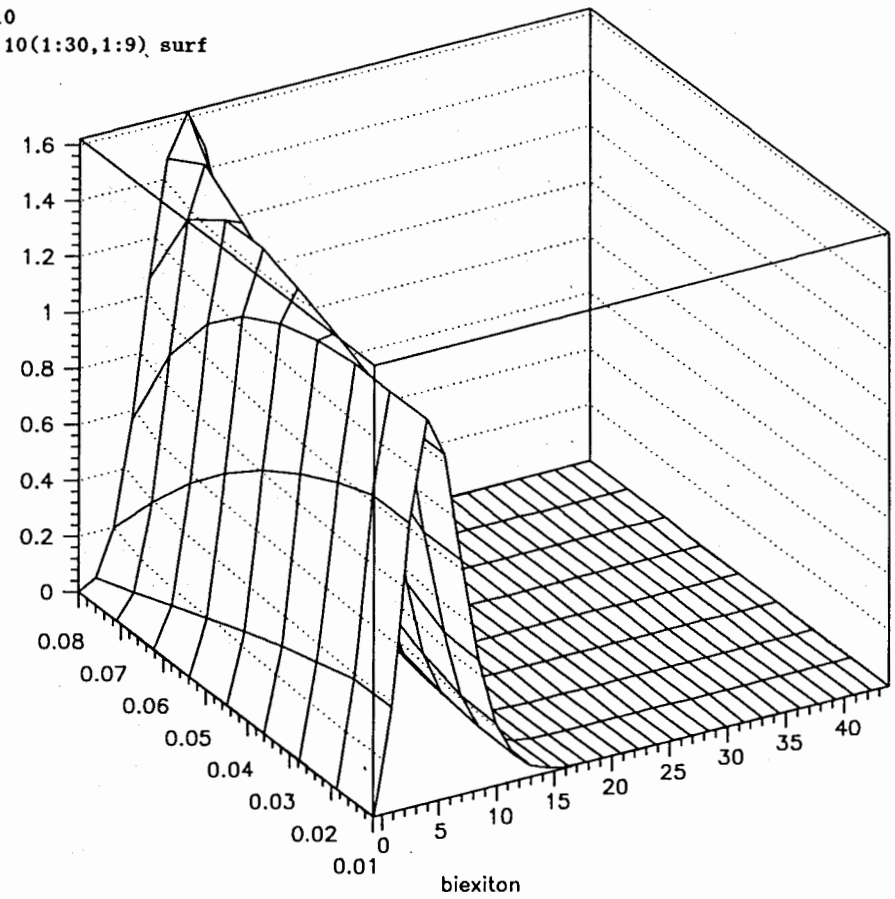

Рис. 7: Пример построения поверхности

В заключение В.Файн хотел бы выразить свою признательность:

- Ширикову В.П. – за редактирование рукописи и ценные замечания
- Потребениковой Е.Н. – за плодотворное сотрудничество
- Андриянову Ю.И. и Емелину И.А. – за обеспечение надежной работы ЭВМ комплекса КОНТРАСТ
- Иванченко И.М. – за полезные дискуссии
- Виницкому С.И. – за постановку задачи и плодотворное сотрудничество

и всем тем пользователям IBM PC ОИЯИ, которые рискнули своим личным временем и взяли на себя труд по освоению новой неотлаженной системы.

Если у Вас есть вопросы или замечания по работе системы, посылайте их по системе электронной почты по адресу: **fine@main1.jinr.dubna.su**.

Литература

[1] Vandoni C., Zanarini P., Brun R., Coute O. *PAW – Physics Analysis Workstation*, 1992. CERN Program library, Q121, Geneva, 1992.

[2] Kunz P.F. Physics analysis tools. In *Proceeding of the International Conference on Computing in High Energy Physics '91*, 1991, p.303.

[3] Marquina M., Brun R., Carminati F. Experience in managing a large scientific library. In F.Abe Y.Watase, editor, *Proceeding of the International Conference on Computing in High Energy Physics '91*, 1991, p. 315.

[4] Kadantsev S., Fine V. Porting PAW on PC's under MS DOS. In *Second International Workshop on Software Engineering, Artificial Intelligence and Expert Systems for High Energy And Nuclear Physics*, 1992.

[5] В.К.Балашов. Машинная графика на UNIX машинах ЛВТА ОИЯИ. Сообщение ОИЯИ Р10-91-532, Дубна, 1991

[6] O.Couete. HIGZ In *CERN Computer Newsletter* N. 207, CERN, Geneva, 1992.

[7] Lienart D., Brun R. *HBOOK User Guide - Version 4.* Y250 CERN Program Library, 1988.

[8] Brun R., Coute O., Cremel N. *HPLOT User Guide - Version 5.*, Y251. CERN Program Library, 1988.

[9] James F. and Roos M. *MINUIT - Function Minimization and Error Analysis*, D506. CERN Program Library, 1989.

[10] Brun R. and Zanarini P. KUIP – kit for a user interface package. I102. CERN Program Library, 1991.

[11] Г.Эндерле и др. Международный стандарт GKS, М.:Радио и связь, 1988.

[12] James F. and Roos M. *MINUIT - Function Minimization and Error Analysis*, D506. CERN Program Library, 1989.

[13] *NDP Fortran Reference Manual*, MicroWay, Inc. 1992.

[14] Bock R., Brun R. et al *HIGZ - High level Interface to Graphics and ZEBRA*, Q120. CERN Program Library, 1988.

[15] L.Lamport. LaTeX: *A Document Preparation System*. Addison-Wesley Publishing Company, 1986.

[16] M.Goossens. 4.2. Merging PostScript Pictures with Text in LaTeXIn *CERN Computer Newsletter*, N 207, CERN, 1992.

[17] Говорун Н.Н. и др. Локальная вычислительная сеть ОИЯИ: аппаратное и программное обеспечение Сообщение ОИЯИ Д11-86-702, Дубна, 1986

[18] Краснослободцев В.И. и др. Организация связи персональных ЭВМ типа IBM XT/AT с ЕС ЭВМ через устройство группового управления ЕС-7922 Сообщение ОИЯИ Р10-89-841, Дубна, 1989

[19] Потребеникова Е.В., Файн В.Э. Организация взаимодействия рабочей станции на базе PC с ЕС ЭВМ. *Информационный бюллетень ЛВТА*, 4-5699, Дубна, 1992, с.36.

[20] Виницкий С.И., Мардоян Л.Г., Погосян Г.С., Сисакян А.Н., Стриж Т.А. Сообщение ОИЯИ Р2-92-302, Дубна, 1992.

Смирнова О.Г., Стриж Т.А., Файн В.Э.
Р11-92-416

Использование системы PAW в MS DOS

Дано общее представление о системе PAW (Physics Analysis Workstation), особенностях ее реализации для персональных ЭВМ типа IBM PC AT386/486 с операционной системой MS DOS. Приводится несколько "живых" примеров, демонстрирующих уникальные возможности системы, которые не нашли достаточного отражения в оригинальном "Руководстве для пользователей" (ЦЕРН).

Работа выполнена в Лаборатории вычислительной техники и автоматизации ОИЯИ.

Сообщение Объединенного института ядерных исследований. Дубна 1992

Перевод авторов

Smirnova O.G., Strizh T.A., Fine V.E.
P11-92-416

PAW Using under MS DOS

The only simple knowledge about the PAW — Physics Analysis Workstation and the special features of its MS DOS implementation is presented. To show the PAW system facilities, some "vital" examples are performed. This is an addition to the original CERN PAW User Guide.

The investigation has been performed at the Laboratory of Computing Techniques and Automation, JINR.

Communication of the Joint Institute for Nuclear Research. Dubna 1992