УДК 004.438

Е. Р. Алексеев, П. А. Демин, Д. А. Костюк

ВОЗМОЖНОСТИ ГРАФИЧЕСКОГО ВЫВОДА РЕЗУЛЬТАТОВ В ПОСЛЕДОВАТЕЛЬНЫХ И ПАРАЛЛЕЛЬНЫХ КРОССПЛАТФОРМЕННЫХ ВЫЧИСЛИТЕЛЬНЫХ ПРИЛОЖЕНИЯХ НА ФОРТРАНЕ И С(С++)

Рассмотрено применение сторонних средств построения графиков в кроссплатформенных консольных приложениях, написанных на языках Фортран и С/С++. Обосновывается выбор современных диалектов данных языков для математических вычислений. Проанализировано использование пакета gnuplot для визуализации результатов расчетов; рассматриваются различные варианты его взаимодействия с программой пользователя. Особое внимание уделено использованию потоков С++ для взаимодействия с программой gnuplot для графического вывода результатов. Описана библиотека gnufor2, которая является интерфейсом между компиляторами Фортрана и gnuplot. Рассмотрена визуализация результатов математических вычислений с помощью пакета dislin в программах на С(С++) и Фортране. Впервые на русском языке подробно описаны основные возможности и функции пакета. Представлены реальные примеры графического вывода результатов в кроссплатформенных консольных программах на С(С++) и Фортране.

Ключевые слова: консольное приложение, компилятор, графические библиотеки, g++, gfortran, ifort, gnuplot, dislin.

На современном этапе развития технологий возрастает роль вычислительной математики и математического моделирования при решении современных инженерных и технических задач. Современному человеку важно не просто уметь решать сложные математические и технологические задач, но и получать их численное решение в течении определённого времени. Возрастает необходимость разработки высокоэффективных вычислительных программ решения

сложных научно-технических задач. Во многих отраслях разрабатываются параллельные приложения решения подобных задач. Основными языками для разработки последовательных и параллельных вычислительных приложений являются Фортран и C(C++).

Первый язык программирования Фортран в последнее десятилетие испытывает второе рождение. Разрабатываются новые стандарты языка, появляются новые версии высокоэффективных кроссплатформенных компиляторов. Современный язык программирования Фортран представляет широкие возможности для разработки вычислительных приложений. Стандарт языка позволяет использовать конвейерные, матричные операции, последовательные и параллельные циклы. Большинство компиляторов являются кроссплатформенными и поддерживают все современные технологии параллельного программирования: MPI, OpenMP, многопоточность. Кроме того, в последний стандарт Фортрана (Fortran-2008) поддержка параллельного программирования в виде комассивов (coarrays) включена на уровне языка [1]. Вычислительные программы, разрабатываемые на Фортране, являются одними из самых быстрых[1] [2]. Для переноса программ с одной платформы на другую практически не требуется модификация кода. Платой за эту универсальность является отсутствие поддержки графического вывода на уровне языка. Все современные компиляторы языка (gfortran, ifort и др.) ориентированы на разработку высокоэффективных последовательных и параллельных консольных приложений.

Вывод результатов исследований в графическом виде является неотъемлемой частью большинства вычислительных программ. Не смотря на отсутствие поддержки работы с графикой на уровне языка и кроссплатформенных компиляторов, существует несколько независимых свободных и проприетарных приложений, которые можно использовать для графического вывода в программах, разрабатываемых на Фортране.

[1] Программы на языке С по быстродействию могут составить конкуренцию программам на Фортране при решении некоторых вычислительных задач. Однако при разработке параллельных и последовательных программ обработки массивов и матриц у Фортрана сегодня нет конкурентов.

Язык программирования C(C++) также широко применяется при решении вычислительных задач. Большинство современных компиляторов (MS Visual Studio, icpc, gcc(g++) и др.) поддерживают все современные технологии параллельного программирования: MPI, OpenMP, многопоточность. Современные среды разработки (MS Visual Studio [3], Qt Creator[4]) поддерживают не только графический вывод результатов, но и разработку приложений с графическим интерфейсом. Однако, при разработке вычислительных приложений с использование подобных IDE возникает другая проблема – разработка кроссплатформенных переносимых вычислительных приложений. Поэтому при разработке приложений на языке C (C++) также необходим простой инструмент, позволяющий разрабатывать кроссплатформенные вычислительные приложения с графическим выводом результатов.

В данной работе будут рассмотрено совместное использование компиляторов Фортрана и C(C++) и программ GNU Plot [5] и dislin [6].

Gnuplot – одна из самых распространённых свободных кроссплатформенных программ для графического вывода результатов. С помощью Gnuplot можно строить высококачественные двух- и трёхмерные графики любой сложности. Подробно о ней можно прочитать на английском [7] или русском [8] языках.

Популярность gnuplot во многом обеспечивается простотой, с которой его могут использовать другие программы для построения графиков по собственным данным. При программировании вычислительных задач – это свойство, по нашему мнению, оказывается полезным начиная с самых простейших задач визуализации простейших математических вычислений. В отличие от специализированных динамически подключаемых библиотек, gnuplot очень широко распространен, и часто уже присутствует в дистрибутиве ОС (так как входит в список зависимостей целого ряда программных инструментов).

Одним из способов вывода графической информации с использованием gnuplot является перенаправление стандартного вывода в поток, в котором работает программа gnuplot. Рассмотрим это на примере.

```
#include <iostream>
#include <cstdio>
#include <math.h>
#ifdef WIN32
 #define GNUPLOT_NAME "pgnuplot -persist"
#else
 #define GNUPLOT_NAME "gnuplot -persist"
#endif
int main()
{
//Создаём поток, в котором будет работать Gnuplot
// С точки зрения пользователя просто создаётся
// графическое окно, в котором с помощью fprintf и
// gnuplot строим график
FILE *gpipe = popen(GNUPLOT_NAME, "w");
// С помощью fprintf , данные отправляются не в файл,
// а в поток Gnuplot
fprintf(gpipe,"set xrange [-20*pi:20*pi]\n");
fprintf(gpipe,"set yrange [-2:2]\n");
fprintf(gpipe,"set samples 1000\n");
fprintf(gpipe,"plot sin(x)\n");
fprintf(gpipe, "exit\n");
// Закрываем поток.
pclose(gpipe);
//Создаём новый поток (графическое окно),
// в котором будет работать Gnuplot
FILE *gpipe1 = popen(GNUPLOT_NAME, "w");
fprintf(gpipe1,
"plot 1/sqrt(2)/sqrt(pi)*exp(-(x-2.35)**2/1.6**2) \n");
```

```
fprintf(gpipe1, "exit\n");
pclose(gpipe1);
FILE *gpipe2 = popen(GNUPLOT_NAME, "w");
fprintf(gpipe2,
"plot [0:40] 20.0*atan(x-20.0) + 32 + sin(x)\n");
fprintf(gpipe2, "exit\n");
pclose(gpipe2);
FILE *gpipe5 = popen(GNUPLOT_NAME, "w");
fprintf(gpipe5,
"x(u,v)= v<pi ? (2.5-1.5*cos(v))*cos(u):");
fprintf(gpipe5,
"v<2*pi ? (2.5-1.5*cos(v))*cos(u):");
fprintf(gpipe5,
"v<3*pi ? -2+(2+cos(u))*cos(v): -2+2*cos(v)-cos(u)\n");
fprintf(gpipe5,
"y(u,v)= v<pi ? (2.5-1.5*cos(v))*sin(u):");
fprintf(gpipe5,"v<2*pi ? (2.5-1.5*cos(v))*sin(u):");
fprintf(gpipe5,"v<3*pi ? sin(u): sin(u)\n");
fprintf(gpipe5,
"z(u,v)= v<pi ? -2.5*sin(v): v < 2*pi ? 3*v-3*pi:");
fprintf(gpipe5,
"v<3*pi ? (2+cos(u))*sin(v)+3*pi: -3*v+12*pi \n");
fprintf(gpipe5,"set parametric\n");
fprintf(gpipe5,"set pm3d explicit\n");
fprintf(gpipe5,"set pal rgb 9,9,3\n");
fprintf(gpipe5,"unset colorbox \n");
fprintf(gpipe5,"unset key\n");
fprintf(gpipe5,"unset border\n");
fprintf(gpipe5,"unset xtics\n");
```

```
fprintf(gpipe5,"unset ytics\n");
fprintf(gpipe5,"unset ztics\n");
fprintf(gpipe5,"set hidden3d\n");
fprintf(gpipe5,"set surface\n");
fprintf(gpipe5,"set ticslevel 0\n");
fprintf(gpipe5,"set size square\n");
fprintf(gpipe5,"set view 60,210,1.5,1\n");
fprintf(gpipe5,"set isosamples 18,48\n");
fprintf(gpipe5,"set xrange[-8:10]\n");
fprintf(gpipe5,"set yrange[-9:9]\n");
fprintf(gpipe5,"set urange[0:2*pi]\n");
fprintf(gpipe5,"set vrange[0:4*pi]\n");
fprintf(gpipe5,"set multiplot\n");
fprintf(gpipe5,
"splot x(u,v),y(u,v),-z(u,v) w pm3d\n");
fprintf(gpipe5,"splot x(u,v),y(u,v),-z(u,v) lt 4\n");
fprintf(gpipe5,"unset multiplot\n");
fprintf(gpipe5, "exit\n");
pclose(gpipe5);
return 0;
}
```

Обращает на себя внимание тот факт, что не нужно никаких библиотек, никаких ключей компиляции, обычная компиляция с помощью g++. Для запуска подобных приложений нужен gnuplot и g++.

Некоторые из окон с графиками, генерируемые программой приведены ниже (рис. 1–2).

Кроме использования перенаправления программа может передавать данные и команды gnuplot для построения графика через временные файлы и средствами конвейеризации в командной строке.

Последний вариант удобен, когда код программы должен быть как можно проще (в демонстрационных целях, для экономии времени или по другим причинам).

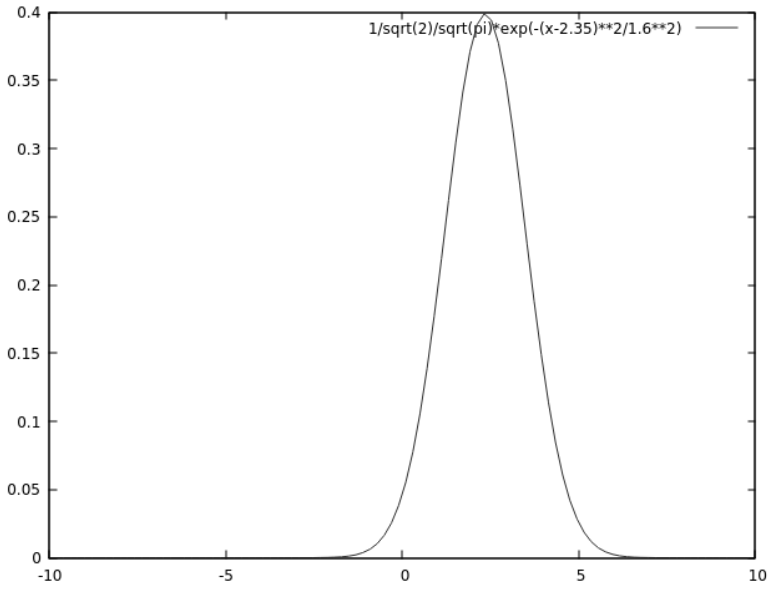


Рис. 1.

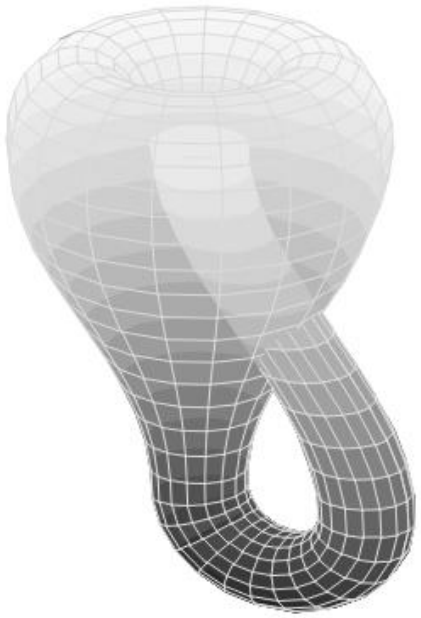


Рис. 2.

Программа, выполняющая вычисления, может просто выводить на стандартный вывод команды для gnuplot и таблицу данных для построения графика. При первом запуске пользователь может визуально оценить печатаемые программой табличные данные. При следующем же запуске пользователь запускает программу совместно с gnuplot, объединив их в так называемый конвейер. Этот подход требует минимум знаний о синтаксисе команд gnuplot: фактически, дается команда, указывающая тип графика, а затем произвольное количество координат точек.

На рис. 3 представлен пример такого решения. Иллюстрация включает три окна: окно редактора с исходным текстом программы, окно терминала с командной строкой ее запуска в конвейере с gnuplot и результирующий график.

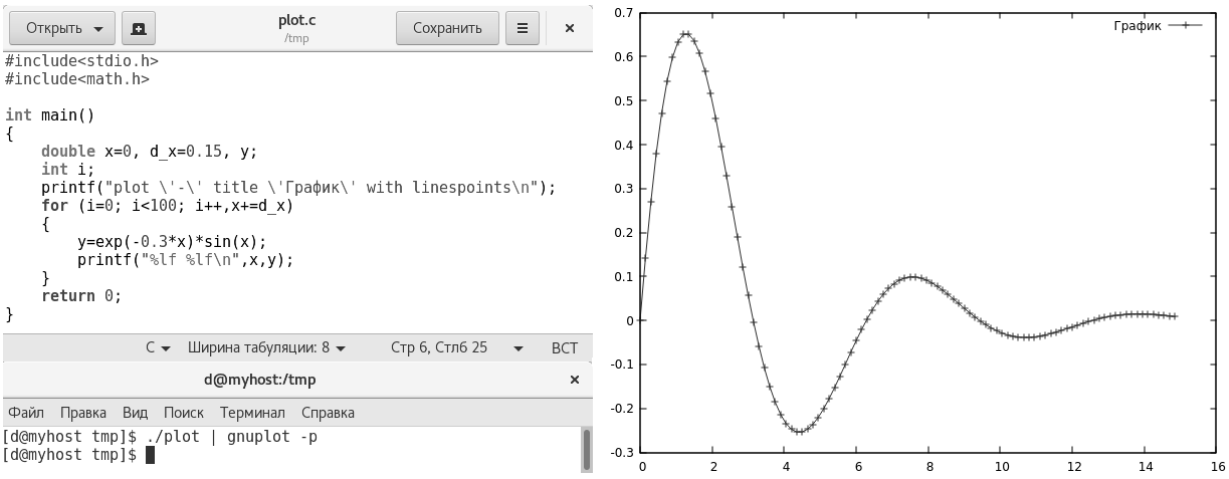


Рис. 3.

Приложение gnufor2 [9] является своеобразным интерфейсом между программой на Фортране и Gnuplot. Своё приложение автор распространяет под свободной лицензией GNU GPL. В состав пакета входит модуль gnufor2 и разработанный автором несложный makefile для подключения модуля к вашему приложению. В системе должен быть установлен пакет gnuplot. По умолчанию для компиляции приложения используется gfortran, который может быть изменён. Для этого необходимо изменить файл Makefile.

Ниже приведен код на Фортране, который строит график функции с использованием библиотеки gnufor2.

```
program primer
use gnufor2
implicit none
integer, parameter :: N=500
real(8)                 :: x(N), y(N)
integer                         :: i
do i=1,N
                x(i)=10.0*i/N
end do
y=sin(x)+1.0/3*sin(3*x)+1/5.0*sin(5*x)+1.0/7*sin(7*x)
print *,'Пример графика функции'
print *,'call plot(x,y)'
call plot(x,y)
print *,'Нажмите ENTER '
read *
end program primer
```

Результатом работы программы является окно gnuplot с графиком функции (см рис. 4).

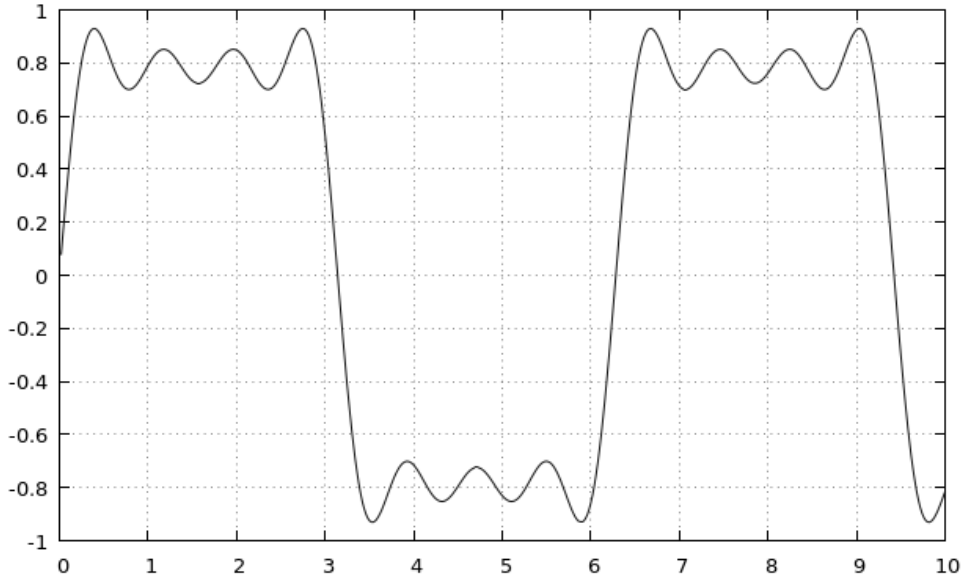


Рис. 4.

Dislin – проприетарная (бесплатная для некоммерческого использования) программа построения высококачественных двух- и трёхмерных графиков, разработанная в институте Макса Планка [10]. К особенностям Dislin относится возможность использования в программах на C(C++), C#, Фортран, Java, Perl, Python, R, Ruby, Julia, Basic.

Рассмотрим, как можно использовать Dislin в консольных приложениях на Фортране и C(C++) в операционных системах семейства Linux. На странице загрузки [11] пакета Dislin можно скачать актуальную версию пакета. Особенности установки описаны в [12, 13].

Для использования графических возможностей Dislin необходимо подключить библиотеку стандартным образом в любом консольном приложении на C(C++) (#include "dislin.h") и Фортране (use dislin).

Рассмотрим основные подпрограммы библиотеки, необходимые для построения графиков. Более подробное описание можно найти в [14] или на странице [15].

Инициализация режима вывода графики осуществляется с помощью подпрограммы disini.

call disini (Фортран)

void disini (); (C)

Выход из режима вывода графики можно осуществить с помощью подпрограммы disfin.

call disfin

void disfin ();

Подпрограмма metafl определяет формат вывода графической информации.

call metafl (cfmt)

void metafl (char *cfmt);

Здесь cfmt – строка, определяющая, куда выводить графическую информацию: на экран ("cons" – полный экран; "xwin" – 2/3 экрана) или в файл (значения

"ps", "eps", "pdf", "svg", "gif", "tif", "png", "bmp" определяют формат выходного файла с графической информацией).

Подпрограмма messag выводит текст в графическом режиме.

call messag (cstr, nx, ny)

void messag (char *cstr, int nx, int ny);

cstr – строка длиной до 256 символов. nx, ny – координаты выводимого текста, относительно левого верхнего угла.

Подпрограмма number выводит вещественное число в графическом окне.

call number (x, ndig, nx, ny)

void number (float x, int ndig, int nx, int ny);

x – вещественное число, ndig – число разрядов в дробной части; если ndig=-1, то x выводится как целое число.

Подпрограмма graf создаёт графические оси для последующего вывода двумерного графика.

call graf (xa, xe, xor, xstep, ya, ye, yor, ystep)

void graf (float xa, float xe, float xor, float xstep,float ya, float ye, float yor, float ystep);

xa, xe, ya, ye – границы изменения переменных X, Y; xor, xstep, yor, ystep – первая метка по осям X, Y соответственно и шаг между метками

Подпрограмма graf3d создаёт трёхмерные графические оси для вывода трёхмерного графика.

call graf3d (xa, xe, xor, xstep, ya, ye, yor, ystep, za, ze, zor, zstep)

void graf3d (float xa, float xe, float xor, float xstep,float ya, float ye, float yor, float ystep, float za, float ze, float zor, float zstep);

xa, xe, ya, ye, za, ze – границы изменения переменных X, Y, Z; xor, xstep, yor, ystep, zor, zstep– первая метка по осям X, Y, Z соответственно и шаг между метками.

Подпрограмма title выводит на экран сформированного подпрограммой titlin заголовка графика

call title

void title ();

Подпрограмма titlin формирует до 4-х строк заголовка графика

call titlin (cstr, n)

void titlin (char *cstr, int n);

cstr – строка (до 132 символов), N – номер строки в заголовке графика.

Подпрограмма ticks определяет количество засечек между линиями сетки

call ticks (ntic, cax)

void ticks (int ntic, char *cax);

ntic – количество засечек между линиями сетки

cax – символ, определяющий ось координат ("x", "y")

Подпрограмма grid определяет количество линий сетки между метками.

call grid (ixgrid, iygrid)

void grid (int ixgrid, int iygrid);

ixgrid, iygridy – количество линий сетки между метками по оси X и Y соответственно, если ixgrid=1, то необходимо выводить линии сетки только в местах меток, если ixgrid=0, то линии сетки не выводить.

Подпрограмма GRID3D определяет количество линий сетки между метками в трехмерном случае.

call grid3d (igrid, jgrid, copt

void grid3d (int igrid, int jgrid, char *copt);

igrid – число линий между метками в направлении X (или в напралении Y для YZ проекциях), jgrid – число линий сетки между метками в направлении Z (или в направлении Y для XY-проекциях).

copt – строка, определяющая какие линии сетки будут нарисованы: "all" – изображение линий сетки в XY-, XZ- и YZ-проекциях, "back" – изображение линий сетки в XZ- YZ-проекциях, "bottom" – изображение линий сетки только в XY-проекции.

Подпрограмма name предназначена для подписи координатных осей.

call name (cstr, cax)

void name (char *cstr, char *cax);

cstr – метка оси, cax – имя оси.

Подпрограмма scrmod определяет фон графического окна.

call scrmod (cmod)

void scrmod (char *cmod);

chmod может принимать следующие значения: "auto" (по умолчанию) использует чёрный в качестве фона окна или графического файла; "revers" изменяет фон на белый; "norev" устанавливает фон, как чёрный, а цвет пера – белый.

Подпрограмма axsbgd определяет цвет фона графика.

call axsbgd (nclr)

void axsbgd (int nclr);

nclr – цвет фона, который может быть сформирован функцией intrgb (например, ic=intrgb (0.95,0.95,0.95);)

Подпрограмма curve выводит график путём соединения точек прямыми линиями.

call curve (x, y, n)

void curve (float *x, float *y, int n);

Массивы x, y содержат X и Y координаты точек. В случае полярного графика – радиусы и углы соответственно, n – количество точек.

Подпрограмма curve может выводить как графики функций, так точечные графики. Способ вывода и соединения точек определяется подпрограммой inmcrk.

call inmcrk (nmrk)

void inmcrk (int nmrk);

nmrk может принимать положительное, отрицательное или нулевое значение. Если nmrk=0, то функция curve соединяет точки прямыми. Если nmrk – от-

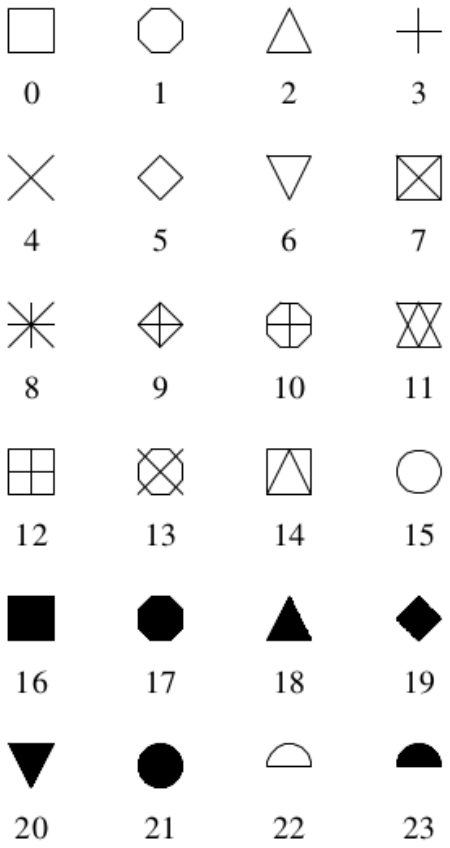
рицательное, то функция curve просто выводит точки на экран (используя определённый символ в качестве точки), не соединяя их. При положительном значении nmrk точки выводятся на экран и соединяются прямыми линиями. Положительное или отрицательное значение n в качестве параметра nmrk выводит на экран каждую n-ю точку с помощью определённого символа.

Символ для изображения точечных графиков определяется подпрограммой marker.

call marker (nsym)

void marker (int nsym);

параметр nsym определяет маркер для изображения точечного графика [16].



Процедура hsymbl определяет размер символа для вывода точечного графика.

call hsymbl (k)

void hsymbl (int k);

Вывод легенд осуществляется в несколько этапов.

1. Определяем буфер легенды – строковую переменную, достаточную для хранения всех строк легенды.

2. Инициализируем легенду с помощью функции legini.

call legini (cbuf, nlin, nmaxln)

void legini (char *cbuf, int nlin, int nmaxln);

cbuf – буфер легенды, nlin – количество строк легенды, nmaxln – максимальная длина строки легенды, должно выполняться выполняться соотношение количество символов в буфере легенды = nlin*nmaxln

3. Формирование строк легенды с помощью подпрограмм legini

call leglin (cbuf, cstr, ilin)

void leglin (char *cbuf, char *cstr, int ilin);

cbuf – строка – буфер легенды, cstr – текст, формирующий очередную подпись легенды, ilin – номер графика, который будет подписан текстом cstr, знгачение ilin должно находится между 1 и nlin.

4. Вывод легенды на экран с помощью подпрограммы legend

call legend (cbuf, ncor)

void legend (char *cbuf, int ncor);

cbuf – сформированный буфер легенды, ncor определяет местоположение легенды на графике: 1 – нижний левый угол; 2 – нижний правый угол; 3 – верхний правый угол; 4 – вехний левый угол; 5 – слева ниже осей координат; 6 – справа ниже осей координат; 7 – справа выше осей координат; 8 – слева выше осей координат.

Подпрограммы surmat и surface рисуют трёх-мерные поверхности функции z=f(x,y). Значения функции должны хранится в виде матрицы; surmat предполагает, что значения функции соответствуют линейной сетке в XY-проекции, в то время как surface может использоваться с нелинейной сеткой.

call surmat (z, ixdim, iydim, ixpts, iypts)

call surfce (x, ixdim, y, iydim, z)

void surmat (float *z, int ixdim, int iydim, int ixpts, int iypts);

void surfce (float *x, int ixdim, float *y, int iydim, float *z);

x, y – массивы X- и Y-координат, z – матрица, хранящая значения функции в точках; xdim, iydim – размерности матрицы z и массивов x и y; ixpts, iypts – число интерполируемых точек между линиями сетки по осям X и Y.

Рассмотрим использование dislin на примере реальных программ на языках C и Fortran.

В «Основах химии» Д. И. Менделеева приводятся данные о растворимости азотнокислого натрия $NaNO_3$ в зависимости от температуры воды. Число условных частей $NaNO_3$, растворяющихся в 100 частях воды при соответствующих температурах, представлено в таблице.

T, °C	0	4	10	15	21	29	36	51	68
n	66.7	71	76.3	80.6	85.7	92.9	99.4	113.6	125.1

Требуется определить растворимость азотнокислого натрия при температурах 18, 27, 35, 47 и 60 градусов в случае линейной зависимости и найти коэффициент корреляции. Параметры линейной зависимости y=a+bx рассчитываются

$$a = \frac{\sum_{i=1}^{n} y_i - b \cdot \sum_{i=1}^{n} x_i}{n},$$

по формулам $b = \frac{n \cdot \sum_{i=1}^{n} y_i \cdot x_i - \sum_{i=1}^{n} y_i \cdot \sum_{i=1}^{n} x_i}{n \cdot \sum_{i=1}^{n} x_i^{\,2} - \left(\sum_{i=1}^{n} x_i \right)^2}$. Коэффициент корреляции вычисляется следующим образом $r = \frac{\sum_{i=1}^{n} (x_i - M_x) \cdot (y_i - M_y)}{\sqrt{\sum_{i=1}^{n} (x_i - M_x)^2 \cdot \sum_{i=1}^{n} (y_i - M_y)^2}}$.

Ниже приведены коды программ решения задачи на Фортране с комментариями, демонстрирующие вывод графиков с помощью библиотеки dislin.

```
program exa_1
! Подключение библиотеки dislin.
 use dislin
 implicit none
 integer, parameter :: n=9
 integer, parameter :: k=5
! Массивы экспериментальных точек.
 real, dimension (n) :: t0,n0
!t1 – массивы температур (18, 27, 35, 47, 60), при
! которыхнеобходимо найти ожидаемое значение
!растворимости(Массив n1).
 real, dimension (k):: t1,n1
! Массивы t2, n2 – определяют линию регрессии.
 real t2(2),n2(2),mt,mn,r,a,b
 character (len=45) :: cbuf
 integer :: i,ic
 print *, "array T"
 read *,t0
 print *, "array n"
 read *,n0
!Среднее значение массива температур.
 mt=sum(t0)/n
!Среднее значение массива растворимостей.
 mn=sum(n0)/n
!Вычисление коэффициента корреляции.
r=sum((t0-mt)*(t0-mt))/sqrt(sum((t0-mt)**2)*sum((t0-mt)**2))
 print *,"r=",r
!Вычисление коэффициентов линии регрессии a и b,
!y=a+bx.
```

```
 b=(n*dot_product(t0,n0)-sum(t0)*sum(n0))/(n*sum(t0*t0)-sum(t0)**2)
 a=(sum(n0)-b*sum(t0))/n
 print *,"a=",a," b=",b
 print *, "array T1"
 read *,t1
!Вычисление ожидаемого значения растворимостей.
 n1=a+b*t1
!Определение точек для изображения линии регрессии.
 t2(1)=t0(1)
 t2(2)=t0(n)
 n2=a+b*t2
 print *,t1
 print *,n1
 print *,t2
 print *,n2
! Начинается блок построения графиков.
! Вывод графика на экран диплея.
 call metafl('xwin')
! Определение белого фона в графическом режиме.
 call scrmod('revers')
! Инициализация графического режима.
 call disini()
! Формирование буфера легенды: 3 строки по 15
! символов.
 call legini(cbuf,3,15)
! Формирование первой строки легенды.
 call leglin(cbuf,'y=a+bx',1)
! Формирование второй строки легенды.
 call leglin(cbuf,'experimental',2)
```

```
! Формирование третей строки легенды.
 call leglin(cbuf,'calculate',3)
! Заголовок легенды.
 call legtit(' ')
 call disalf()
 call axspos(450,1800)
 call axslen(2200,1200)
! Формирование подписей осей X и Y.
 call name('t','x')
 call name('n','y')
! Формирование заголовка графика.
 call titlin('Task of Mendeleev',1)
! Определение цвета графика.
 ic=intrgb(0.95,0.95,0.95)
 call axsbgd(ic)
! Формирование графических осей для графика.
 call graf(-10.0,70.0,0.0,10.0,60.0,130.0,70.0,10.0)
 call setrgb(0.7,0.7,0.7)
! Вывод линий сетки.
 call grid(1,1)
 call color('fore')
! Вывод заголовка графика.
 call title()
 call color('red')
! Определение способа соединения точек кривой.
! Соединение точек прямыми.
 call incmrk(0)
! Вывод линии регрессии.
 call curve(t2,n2,2)
```

```
 call color("blue")
! Определение маркера для вывода отдельных точек
! графика.
 call marker(15)
! Определение размера маркера
 call hsymbl(17)
! Определение способа соединения точек кривой.
! Вывод отдельных точек без соединения их.
! Выводить все точки.
 call incmrk(-1)
! Вывод экспериментальных точек
 call curve(t0,n0,n)
 call color('blue')
! Определение маркера для вывода отдельных точек
! графика.
 call marker(18)
! Определение способа соединения точек кривой.
! Вывод отдельных точек без соединения их.
! Выводить все точки.
 call incmrk(-1)
! Вывод рассчитанных ожидаемых значений в заданных
! точках.
 call curve(t1,n1,k)
! Вывода легенды
 call legend(cbuf,8)
! Выход из режима вывода графики.
 call disfin()
end program exa_1
```

Сформированный программой график представлен на рис. 5.

Пример построения трехмерного графика $Z = 2 \cdot \cos\left(\frac{x}{2}\right) \cdot \cos\left(\frac{y}{3}\right), x \in [0;4\pi], y \in [0;4\pi]$ с использованием библиотеки dislin на языке С представлен на листинге ниже. Сформированный график изображён на рис. 6.

```
#include <stdio.h>
#include <math.h>
#include "dislin.h"
int main ()
{ int n = 100 ,i, j; float z[n][n];
 float dx=4*M_PI / (n-1);
 char *ctit1 = "Surface Plot",
 *ctit2 = "z=2*cos(x/2)*cos(y/3)";
 for (i = 0; i < n; i++)
 for (j = 0; j < n; j++)
 z[i][j] = 2*cos(i*dx/2.0)*cos(j*dx/3.0);
 scrmod ("revers"); metafl ("xwin");
 disini (); disalf ();
 name ("X-axis", "x"); name ("Y-axis", "y");
 name ("Z-axis", "z");
 titlin (ctit1, 2); titlin (ctit2, 4);
 view3d (-5.0, -5.0, 4.0, "abs");
 graf3d (0.0, 15.0, 1.0, 2.0, 0.0, 15.0, 1.0, 2.0,-2.5, 2.5, -2.0, 0.5);
 height (50); title ();
 color ("red");
 surmat ((float *) z, 100, 100, 1, 1);
 disfin ();
 return 0;
}
```

Рассмотренные свободные и проприетарные библиотеки и модули позволяют организовать высококачественный графический вывод результатов в вычислительных приложениях на С и Фортране.

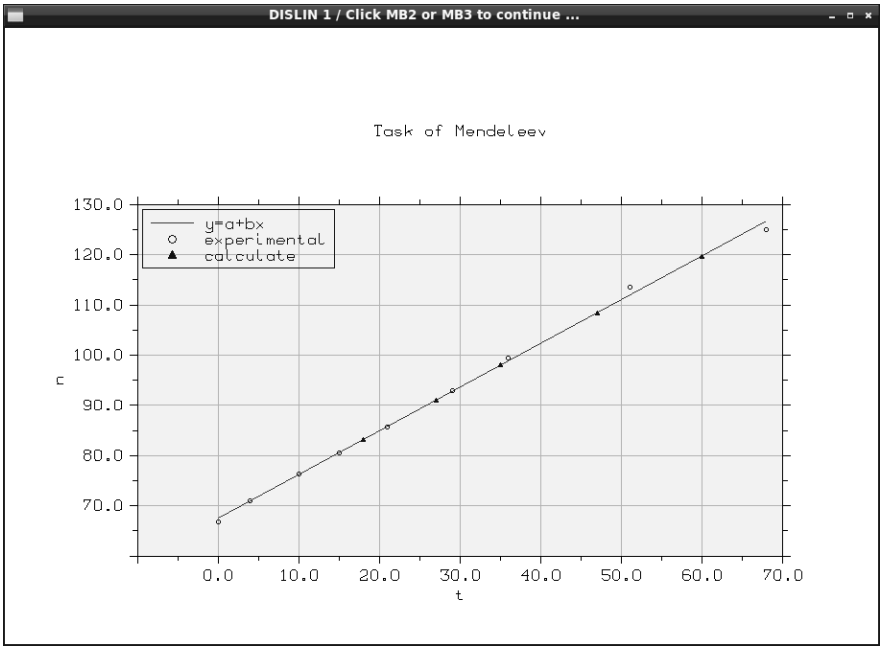


Рис. 5.

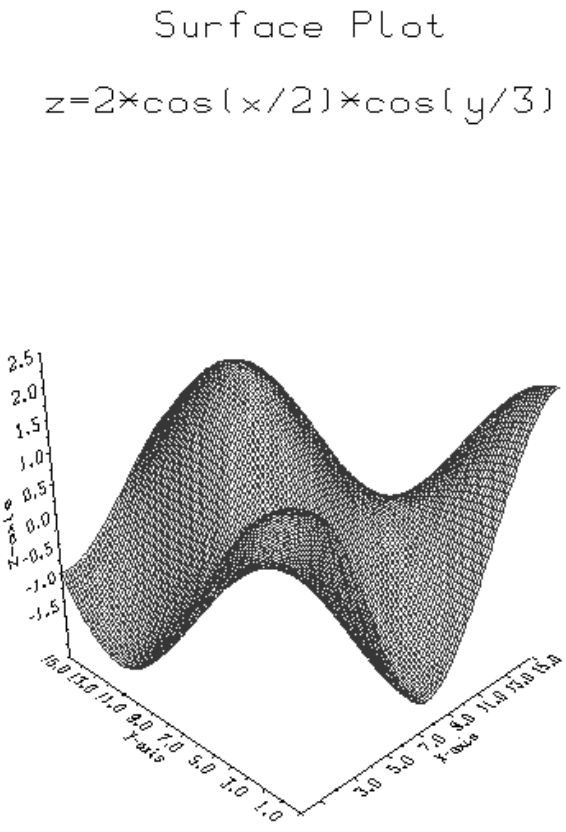


Рис. 6.

Наличие независимых графических библиотек и мощных компиляторов языка Фортран (ifort, gfortran) делают его серьёзным конкурентом существующим сегодня системам математического моделирования.

Список литературы

1. *Горелик А. М.* Программирование на современном Фортране. М.: Финансы и статистика, 2006.

2. *Алексеев Е. Р., Соболева О. В.* Современный язык программирования фортран в образовании и научных исследованиях // Современные информационные технологии и ИТ-образование: сб. материалов междунар. ежегод. науч.-практ. конф., 2016. Т. 12. № 4. С. 110–116.

3. *Алексеев Е. Р.* Программирование в Microsoft Visual C++ и Turbo C++ Explorer / под общ. ред. О. В. Чесноковой. М.: НТ Пресс, 2007.

4. Программирование на языке C++ в среде qt Creator: / Е. Р. Алексеев, Г. Г. Злобин., Д. А. Костюк, О. В. Чеснокова, А. С. Чмыхало. М.: ALT Linux, 2015.

5. gnuplot homepage. URL: http://www.gnuplot.info (дата обращения: 07.02.2017).

6. DISLIN | Max Planck Institute for Solar System Research. URL: http://www.mps.mpg.de/dislin (дата обращения: 07.02.2017).

7. Official gnuplot documentation. URL: http://www.gnuplot.info/documentation.html (дата обращения: 07.02.2017).

8. *Шнейвайс А. Б.* Азы GNUPLOTa. URL: http://www.astro.spbu.ru/sites/default /files/gnuazu6_0.pdf (дата обращения: 07.02.2017).

9. GNUFOR2 – Gnuplot/Fortran interface. URL: http://www.math.yorku.ca/~akuznets/gnufor2/ (дата обращения: 07.02.2017).

10. Home | Max Planck Institute for Solar System Research. URL: http://www.mps .mpg.de/en (дата обращения: 07.02.2017).

11. Downloads | Max Planck Institute for Solar System Research. URL: http://www.mps.mpg.de/dislin/downloads (дата обращения: 07.02.2017).

12. Install DISLIN 10.4 on Ubuntu 14.04 Trusty Tahr. URL: http://blog.hani-ibrahim.de /en/install-dislin-on-ubuntu.html (дата обращения: 07.02.2017).

13. Install DISLIN 10.6 on Ubuntu 16.04 Xenial Xerus. URL: http://blog.hani-ibrahim.de /en/install-dislin-10-6-on-ubuntu-16-04.html (дата обращения: 07.02.2017).

14. D I S L I N 10.6. A Data Plotting Library by Helmut Michels.

15. Contents. URL: http://www2.mps.mpg.de/dislin/contents.html (дата обращения: 07.02.2017).

16. Symbols | Max Planck Institute for Solar System Research. URL: http://www.mps.mpg.de/1758035/exa_symb (дата обращения 07.02.2017).

АЛЕКСЕЕВ Евгений Ростиславович – кандидат технических наук, профессор кафедры фундаментальной информатики и прикладной математики, Вятский государственный университет. 610000, г. Киров, ул. Московская, 36.

E-mail: er_alekseev@vyatsu.ru

КОСТЮК Дмитрий Александрович – кандидат технических наук, доцент, УО «Брестский государственный технический университет». 224017, Беларусь, г. Брест, ул. Московская улица, 267.

E-mail: dmitriykostiuk@gmail.com

ДЕМИН Петр Александрович – магистрант II курса кафедры фундаментальной информатики и прикладной математики, Вятский государственный университет. 610000, г. Киров, ул. Московская, 36.

E-mail: demin-rabota@ya.ru