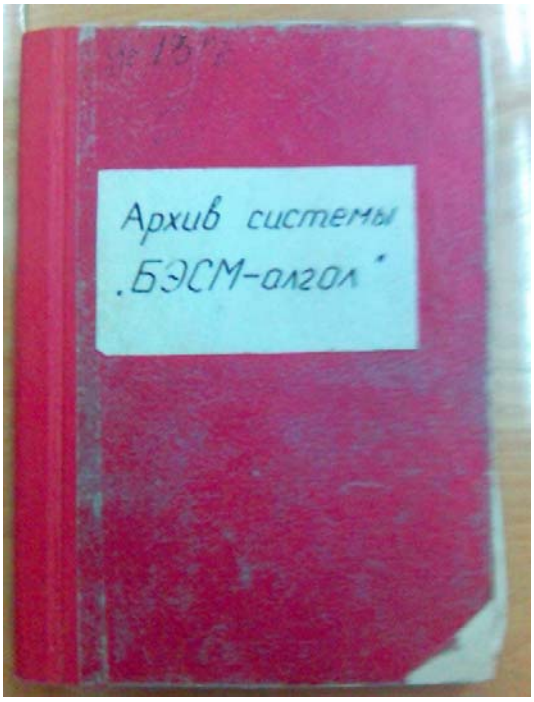
Oldies... But Goldies!

БИБЛИОТЕКА МАТЕМАТИЧЕСКИХ ФУНКЦИЙ

Для системы программирования на языке PL/1 для Windows-XP



Москва 2000

ВЕДЕНИЕ

Данный набор математических подпрограмм был взят из стандартной библиотеки «БЭСМ-АЛГОЛ» образца 1971 года.

История перевода этих подпрограмм на персональный компьютер такова:

Когда происходил массовый переход на персональные компьютеры в начале 90-х годов и старые системы программирования вместе с машинами БЭСМ-6 прекращали свое функционирование, стало жалко просто выбрасывать имеющиеся в стандартной библиотеке программы на языке Алгол. А поскольку при вызове стандартной процедуры в «БЭСМ-АЛГОЛ» вызывался и исходный текст этой стандартной процедуры, была написана «прощальная» программа на Алголе, просто вызывающая все имеющиеся процедуры из библиотеки. Полученный таким образом по существу исходный текст библиотеки был переписан с магнитной ленты на дискеты. Физически возможности включать БЭСМ-6 больше уже не было.

«Исходный» алгольный текст на дискете долго валялся без дела, пока пришедшей в отдел молодой специалистке не поручили привести его в порядок и формально перевести с языка Алгол на язык PL/1, который и использовался тогда на персональных компьютерах.

Перевод с языка на язык осуществлялся простой глобальной заменой «REAL» на «FLOAT», «INTEGER» на «FIXED» и т.п. с последующим ручным исправлением некоторых конструкций типа условных присваиваний. Некоторые алгоритмы, зависящие от архитектуры БЭСМ-6, вроде датчика случайных чисел, основанного на переполнении разрядной сетки, вероятно, будут работать не так, как задумано. Были оставлены все оригинальные имена.

Несмотря на хихиканье остальных сотрудников, эта работа была проведена, включая перевод «наивных» простых программ типа подсчета суммы массива.

В дальнейшем наличие под рукой этой «знакомой» математической библиотеки, а самое главное, наличие исходных текстов разных математических методов оказалось очень полезным и потраченное время было вполне оправдано.

Хотя в последующие годы отдел использовал лишь 6-7 процедур, исходные тексты которых прямо вставлялись в программы и даже дорабатывались, было сэкономлено огромное время на поиски в справочниках и «первичную» отладку.

Главное в этих алгоритмах, на наш взгляд, следующее:

- они потребовались в реальных расчетах, следовательно, гарантированно имеют практическую пользу;
- их составляли программисты, может быть тогда не очень хорошо разбиравшиеся в языках программирования, но зато, как правило, очень хорошо знающие математику;
- у многих алгоритмов есть ссылка на источник их появления, что экономит много времени по сравнению с самостоятельным поиском решений в справочниках и даже в Интернете.

В настоящее время существует множество математических библиотек в разных системах программирования, однако в большинстве из них исходные тексты не доступны и, таким образом, исправить/дополнить какой-либо алгоритм невозможно.

В целом же, как показал опыт использования этой библиотеки, разработки и мысли программистов старшего поколения (в те времена прикладных программистов называли просто математиками) с успехом могут быть использованы.

Именно поэтому представляется важным и полезным и в дальнейшем иметь возможность использовать эту библиотеку. Как это и принято сейчас, она оформлена в виде файла библиотеки MATH.L86 и файла описаний MATH.DCL, однако доступны и все исходные тексты. Каждая подпрограмма названа именем раздела из данного описания, например, метод или алгоритм, описанный в разделе 4.9. будет называться A0409.PL1.

Поскольку с программной точки зрения исходные тексты несложны и невелики по размеру, их легко вставить прямо в свою программу, заменить «ужасные» имена, придуманные во времена Алгола, более разумными и т.п. Все исходные тексты сделаны для чисел с обычной точностью (FLOAT(24)), однако в каждой процедуре описана переименованная константа Ч, заменив которую на 53 можно получить алгоритм с удвоенной точностью (FLOAT(53)).

При использовании библиотеки, в программу нужно вставить оператор

```
%INCLUDE 'MATH.DCL';
```

и после директивы трансляции

```
PLINK <имя программы>.PL1
```

выполнить директиву редактора связи

```
PLINK <имя программы>, MATH.L86[S]
```

1. СПЕЦИАЛЬНЫЕ ФУНКЦИИ

1.1. Процедура-функция вычисления полярного угла

Функция ARG(X,Y) вычисляет полярный угол точки X, Y. Значения ARG лежат в интервале -π < ARG <= π . Для точки (0,0) выдается значение 0.

Описание в PL/1:
DCL ARG ENTRY(FLOAT,FLOAT) RETURNS(FLOAT);

Пример обращения: УГОЛ=ARG(X,Y);

1.2. Функция Бесселя нулевого порядка от вещественного аргумента

Y0(X,M) - процедура-функция, определена для X>0.
X - аргумент функции;
M - код ошибки, возвращает 1, если X<=0, в этом случае значение функции равно 0; или 0, если X>0.

Описание в PL/1: DCL Y0 ENTRY(FLOAT,FIXED) RETURNS(FLOAT);

Пример обращения: F0=Y0(X,M1);

1.3. Функция Бесселя первого порядка от вещественного аргумента

Y1(X,M) - процедура-функция, определена для X>0.
X - аргумент функции;
M - код ошибки, возвращает 1, если X<=0, в этом случае значение функции равно 0; или 0, если X>0.

Описание в PL/1: DCL Y1 ENTRY(FLOAT,FIXED) RETURNS(FLOAT);

Пример обращения: F1=Y1(X,M1);

1.4. Функция Бесселя нулевого порядка от мнимого аргумента

I0(X) - процедура-функция определена для любого аргумента.
X - аргумент функции;

Описание в PL/1: DCL I0 ENTRY(FLOAT) RETURNS(FLOAT);

Пример обращения: F0=I0(X);

1.5. Функция Бесселя первого порядка от мнимого аргумента

I1(X) - процедура-функция определена для любого аргумента.
X - аргумент функции;

Описание в PL/1: DCL I1 ENTRY(FLOAT) RETURNS(FLOAT);

Пример обращения: F1=I1(X);

1.6. Функция Бесселя от вещественного аргумента

J0(X) - процедура-функция определена для любого аргумента.
X - аргумент функции;

Описание в PL/1: DCL J0 ENTRY(FLOAT) RETURNS(FLOAT);

Пример обращения: F0=J0(X);

1.7. Функция Бесселя от мнимого аргумента

J1(X) - процедура-функция определена для любого аргумента.
X - аргумент функции;

Описание в PL/1: DCL J1 ENTRY(FLOAT) RETURNS(FLOAT);

Пример обращения: F1=J1(X);

1.8. Функция Бесселя от мнимого положительного аргумента

K0(X,M) - процедура-функция, определена для X>0.

X - аргумент функции;

M - код ошибки, возвращает 1, если X<=0, в этом случае значение функции равно 0; или 0, если X>0.

Описание в PL/1:

```
DCL K0 ENTRY(FLOAT,FIXED(7)) RETURNS(FLOAT);
```

Пример обращения: F0=K0(X,M1);

1.9. Функция Бесселя от мнимого положительного аргумента

K1(X,M) - процедура-функция, определена для X>0.

X - аргумент функции;

M - код ошибки, возвращает 1, если X<=0, в этом случае значение функции равно 0; или 0, если X>0.

Описание в PL/1:

```
DCL K1 ENTRY(FLOAT,FIXED(7)) RETURNS(FLOAT);
```

Пример обращения: F1=K1(X,M1);

1.10. Процедура вычисления функций Бесселя от вещественного аргумента для последовательности целых порядков

BESSETIN(X,N,EPS,JPOINT);

X - аргумент функций Бесселя;

N - максимальный порядок функций Бесселя. В результате процедуры будут вычислены функции всех целых порядков, начиная с 0 и кончая N, N должно быть неотрицательным числом;

JPOINT - указатель на массив J(0:N) для вычисленных значений функций Бесселя всех целых порядков от 0 до N для данного аргумента X;

EPS - параметр, обуславливающий абсолютную точность вычисления функций Бесселя;

Вычисление производится с помощью применения рекурсивной формулы в обратном направлении от N до 0, для увеличения точности рекурсия повторяется с большим порядком N до тех пор, пока разность между результатами двух последних рекурсий не станет меньше EPS. Учтено замечание Дж. Стеффорда о защите от переполнения.

Описание в PL/1: DCL BESSETIN ENTRY(FLOAT,FIXED,FLOAT,PTR);

Пример обращения: CALL BESSETIN(X,N,EPS,JPOINT);

Источник: М.И. Агеев. Алгоритмы (1-50).

1.11. Гамма-функция

GAMMA(X1,M);

X1 - аргумент функции;
M - код ошибки, возвращает 1, если X>21.427 или X=0, или 0, если все нормально.
Функция вычисляет $\Gamma(x)=\int_0^{\infty} e^{-t} t^{x}\,dt$
Описание в PL/1:
DCL GAMMA ENTRY(FLOAT,FIXED) RETURNS(FLOAT);

Пример обращения: F1=GAMMA(X,M);

1.12. Функция вычисления интегральных синуса и косинуса Френеля

FRESNEL(U,A,B)

U - аргумент функции,
A - интегральный синус
B - интегральный косинус

Описание в PL/1: DCL FRESNEL ENTRY(FLOAT,FLOAT,FLOAT);

Пример обращения: CALL FRESNEL(X,SN,CN);

Источник: М.И. Агеев. Алгоритм 88а.

1.13. Функция вычисления интегрального синуса Френеля

FRSIN(U)

U - аргумент функции

Описание в PL/1: DCL FRSIN ENTRY(FLOAT) RETURNS(FLOAT);

Пример обращения: SN=FRSIN(X);

Источник: М.И. Агеев. Алгоритм 89а.

1.14. Функция вычисления интегрального косинуса Френеля

FRCOS(U)

U - аргумент функции;

Описание в PL/1: DCL FRCOS ENTRY(FLOAT) RETURNS(FLOAT);

Пример обращения: CS=FRCOS(X);

Источник: М.И. Агеев. Алгоритм 90а.

1.15. Функция кругового значения ARCSIN(X) в интервале (0,2π)

ARCSINCIRC(X,Y)

X - аргумент функции,
Y - любое число со знаком косинуса искомого угла.

Описание в PL/1:
DCL ARCSINCIRC ENTRY(FLOAT,FLOAT) RETURNS(FLOAT);

Пример обращения: AS=ARCSINCIRC(X,Y);

1.16. Функция кругового значения ARCTG(X) в интервале (0,2π)

ARCTGCIRC(X,Y)

X - аргумент функции
Y - любое число со знаком косинуса искомого угла.

Описание в PL/1:
DCL ARCTGCIRC ENTRY(FLOAT,FLOAT) RETURNS(FLOAT);

Пример обращения: TG=ARCTGCIRC(X,Y);

1.17. Функция вычисления интеграла вероятности

EFR(X)
X - аргумент функции

$$erf(x)= 2/sqrt(\pi)* \int_{0}^{x} (e)**(-1u**2)\ du$$

Описание в PL/1: DCL ERF ENTRY(FLOAT) RETURNS(FLOAT);

Пример обращения: F=ERF(X);

2. ИНТЕГРИРОВАНИЕ

2.1. Функция вычисления определенного интеграла методом Симпсона с заданной абсолютной погрешностью

SIMPSON(A,B,F,EPS)

A и B - границы интеграла
F - имя процедуры-функции
EPS - абсолютная погрешность результата

Процедура может быть использована рекурсивно для вычисления кратных интегралов. В этом случае пределы интегрирования должны быть константами.

Пример: вычислить интеграл: $\int_0^2 dy \int_0^2 x\,dx$

Программа на PL1:

```
S: PROC OPTIONS(MAIN);
DCL R              FLOAT;
DCL SIMPSON ENTRY(FLOAT,FLOAT,ENTRY(FLOAT)
                              RETURNS(FLOAT),FLOAT)
                              RETURNS (FLOAT);
F1: PROC(X) RETURNS(FLOAT); DCL X FLOAT; RETURN(X); END;
F2: PROC(X) RETURNS(FLOAT); DCL X FLOAT;
                 RETURN(SIMPSON(0,2,F1,0.001)); END;
START:
R=SIMPSON(0,2,F2,0.001);
PUT SKIP LIST('R=',R);
END;
```

Описание в PL/1:

```
 DCL SIMPSON ENTRY(FLOAT,FLOAT,
                              ENTRY(FLOAT) RETURNS(FLOAT),
                              FLOAT) RETURNS(FLOAT);
```

Пример обращения: INT=SIMPSON(X1,X2,FUN,EPS);

Источник: "Communication of the ACM". Алгоритм 103.

2.2. Функция вычисления определенного интеграла методом Симпсона с заданной относительной погрешностью

SIMPSON1(F,A,B,EPS)

A и B - границы интеграла
F - имя процедуры-функции
EPS - относительная погрешность результата

Процедура-функция приближенно вычисляет интеграл от функции F в пределах от A до B с относительной погрешностью EPS. Подпрограмма выполняет интегрирование быстрее, чем это делается другими методами.

Процедура может быть использована рекурсивно для вычисления кратных интегралов. В этом случае пределы интегрирования должны быть константами.

Пример: вычислить интеграл: $\int_0^2 dy \int_0^2 x\,dx$

Программа на PL1:

```
S: PROC OPTIONS(MAIN);
DCL R              FLOAT;
DCL SIMPSON1 ENTRY(FLOAT,FLOAT,ENTRY(FLOAT)
                                RETURNS(FLOAT),FLOAT)
                                RETURNS (FLOAT);
F1: PROC(X) RETURNS(FLOAT); DCL X FLOAT; RETURN(X); END;
F2: PROC(X) RETURNS(FLOAT); DCL X FLOAT;
                RETURN(SIMPSON(0,2,F1,0.001)); END;
START:
R=SIMPSON1(0,2,F2,0.001);
PUT SKIP LIST('R=',R);
END;
```

Описание в PL/1: DCL SIMPSON1 ENTRY(ENTRY(FLOAT)
RETURNS(FLOAT), FLOAT,FLOAT,FLOAT) RETURNS(FLOAT);

Пример обращения: IN=SIMPSON1(FUN,X1,X2,EPS);

Источник: М.И. Агеев. Алгоритм 182а.

2.3. Процедуры-функции вычисления определенного интеграла методом Симпсона

SIMPSONF(A,B,XPTR,F,EPS)

Эта процедура предназначена для вычисления определенных интегралов (как однократных, так и многократных) с заданной абсолютной точностью результата.

Пределы интегрирования A и B описаны как процедуры-функции, что позволяет вычислять любые многомерные интегралы.

A и B - границы интеграла
XPTR - указатель на переменную интегрирования
F - имя интегрируемой процедуры-функции
EPS - абсолютная точность результата

Описание в PL/1:

```
 DCL SIMPSONF ENTRY(ENTRY RETURNS(FLOAT),
                    ENTRY RETURNS(FLOAT),
                    PTR, ENTRY RETURNS(FLOAT),FLOAT)
                    RETURNS(FLOAT);
```

Пример обращения: Вычислить интеграл $\int_0^1 dx \int_{x^{**}2}^{sqrt(x)} (y^{**}2+x)\, dy$

Программа на PL1:

```
T:PROC OPTIONS(MAIN);
DCL (RES,Y,X) FLOAT,
   SIMPSONF  ENTRY(ENTRY RETURNS(FLOAT),
                   ENTRY RETURNS(FLOAT),
                   PTR,ENTRY RETURNS(FLOAT),FLOAT)
             RETURNS(FLOAT);

A: PROC RETURNS(FLOAT); RETURN(0); END;
B: PROC RETURNS(FLOAT); RETURN(1); END;
F: PROC RETURNS(FLOAT);
        RETURN(SIMPSONF(AA,BB,ADDR(Y),FF,1.E-2)); END;

AA: PROC RETURNS(FLOAT);  RETURN(X*X); END;
BB: PROC RETURNS(FLOAT);  RETURN(SQRT(X)); END;
FF: PROC RETURNS(FLOAT);  RETURN(Y*Y+X); END;
```

```
START:
RES=SIMPSONF(A,B,ADDR(X),F,1.E-2);
PUT SKIP LIST('ИНТЕГРАЛ=',RES);
END;
```

Результат равен 33/140 или 2.357234E-01

2.4. Вычисление интегралов с заданным числом интервалов

SIMPS1(A,B,XPTR,N,F)

Эта процедура предназначена для вычисления определенных интегралов (как однократных, так и многократных) с заданным числом интервалов разбиения отрезка интегрирования. Пределы интегрирования A и B описаны как процедуры-функции что позволяет вычислять любые многомерные интегралы.

A и B - границы интеграла
XPTR - указатель на переменную интегрирования
N - число интервалов разбиения отрезка [a,b] (число N должно быть четным)
F - имя интегрируемой процедуры-функции

Описание в PL/1:

```
DCL SIMPS1 ENTRY(ENTRY RETURNS(FLOAT),
                        ENTRY RETURNS(FLOAT),PTR,
                        FIXED,ENTRY RETURNS(FLOAT))
                 RETURNS(FLOAT);
```

Пример обращения: Вычислить интеграл $\int_0^1 dx \int_{x**2}^{sqrt(x)} (y**2+x)\, dy$

Программа на PL1:

```
T:PROC OPTIONS(MAIN);
DCL (RES,Y,X) FLOAT,
      SIMPS1   ENTRY(ENTRY RETURNS(FLOAT),
                        ENTRY RETURNS(FLOAT),
                        PTR,FIXED,ENTRY RETURNS(FLOAT))
                 RETURNS(FLOAT);
```

```
A: PROC RETURNS(FLOAT); RETURN(0); END;
B: PROC RETURNS(FLOAT); RETURN(1); END;
F: PROC RETURNS(FLOAT);
        RETURN(SIMPS1(AA,BB,ADDR(Y),20,FF)); END;

AA: PROC RETURNS(FLOAT);  RETURN(X*X); END;
BB: PROC RETURNS(FLOAT);  RETURN(SQRT(X)); END;
FF: PROC RETURNS(FLOAT);  RETURN(Y*Y+X); END;

START:
RES=SIMPS1(A,B,ADDR(X),20,F);
PUT SKIP LIST('ИНТЕГРАЛ=',RES);
END;
```

Результат равен 33/140 или 2.357234E-01

Для вычисления многомерного интеграла процедура SIMPS1 часто оказывается более удобной и более быстродействующей, чем SIMPSONF в силу того, что задаваемая в SIMPSONF абсолютная погрешность EPS для внутренних интегралов должна зависеть от внешних переменных интегрирования (если требуется сократить время счета). На практике подобрать такую зависимость весьма трудно. Если ABS(A-B)<E-18, то происходит немедленный выход из процедуры с выдачей результата, равного 0.

2.5. Процедура вычисления синус- и косинус-преобразований Фурье методом Филона

FILON(F,A,B,P,EPS1,EPS2,YS,YC)

Процедура производит одновременно вычисления двух интегралов:

$$YS=\int_A^B F(X)*SIN(P*X)\,dx \quad \text{и} \quad YC=\int_A^B F(X)*COS(P*X)\,dx$$

A и B - границы интеграла. Верхний предел интегрирования может быть равен бесконечности. Если верхний предел должен быть равен бесконечности, то значение B не должно быть намного больше длины отрезка, на котором функция F(X) практически отлична от 0, при этом процедура будет последовательно вычислять интегралы от A до A+KB, K=1,2,... до достижения заданной точности EPS2 # 0;

F - имя процедуры-функции от которой вычисляется интеграл Фурье

P - заданный аргумент преобразования Фурье

EPS1 - абсолютная точность вычисления интеграла на одном отрезке

EPS2 - абсолютная точность вычисления интеграла на верхнем пределе, равном бесконечности (если верхний предел не равен бесконечности, то фактический параметр EPS2 должен быть равен нулю.

YS YC - выходные параметры

Описание в PL/1:

```
DCL FILON ENTRY(ENTRY(FLOAT) RETURNS(FLOAT),
                FLOAT,FLOAT,FLOAT,FLOAT,FLOAT,FLOAT,FLOAT);
```

Пример обращения: CALL FILON(FUN,X1,X2,P,EPS1,EPS2,FS,FC);

Источник: Ю.Д. Пикин и А.И. Смородик "Алгоритм численного интегрирования осциллирующих функций", сб. Алгоритмы и алгоритмические языки, М., ВЦ АН СССР, 1968, вып.3.

2.6. Процедура вычисления преобразования Фурье методом Филона с заданной относительной точностью результата

FILOTH(F,A,B,P,EPS1,EPS2,YS,YC)

Смысл формальных параметров тот же, что и в процедуре FILON за исключением того, что EPS1 и EPS2 в данном случае представляют собой относительные погрешности, а не абсолютные. Алгоритм вычисления совпадает с приведенным в процедуре FILON. Изменено только условие окончания для достижения заданной точности.

Описание в PL/1:

```
DCL FILOTH ENTRY(ENTRY(FLOAT) RETURNS(FLOAT),
                 FLOAT,FLOAT,FLOAT,FLOAT,FLOAT,FLOAT,FLOAT);
```

Пример обращения: CALL FILOTH(FUN,X1,X2,P,EPS1,EPS2,FS,FC);

2.7. Процедура интегрирования системы дифференциальных уравнений методом Рунге-Кутта на один шаг

RK1STEP(X,YPOINT,H,N,F,X1,Y1POINT)

X - начальное значение независимой переменной;
YPOINT- указатель на массив Y(1:N) решение в точке X;
H - шаг интегрирования;
N - порядок системы;
F - имя процедуры для вычисления правых частей, эта процедура должна быть описана в основной программе как F(X,N,RPOINT,ZPOINT), где X - независимая переменная (первый аргумент правых частей)
N - порядок системы
RPOINT - указатель на массив аргументов R(1:N) для вычисления правых частей
ZPOINT - указатель на массив результатов Z(1:N)
X1 - значение независимой переменной, равное X+H;
YPOINT1- указатель на массив результатов Y1(1:N) решения в точке X1;
Описание в PL/1:

```
DCL RK1STEP ENTRY(FLOAT,PTR,FLOAT,FIXED,
                  ENTRY(FLOAT,FIXED,PTR,PTR),
                  FLOAT,PTR);
```

Пример обращения: CALL RK1STEP(X,YPOINT,H,N,F,X1,Y1POINT);

2.8. Процедура интегрирования системы дифференциальных уравнений модифицированным методом Эйлера на один шаг

EILER1STEP(X,YPOINT,H,N,F,X1,Y1POINT)

X - начальное значение независимой переменной;
YPOINT- указатель на массив Y(1:N) решение в точке X;
H - шаг интегрирования;
N - порядок системы;
F - имя процедуры для вычисления правых частей, эта процедура должна быть описана в основной программе как F(X,N,RPOINT,ZPOINT), где X - независимая переменная (первый аргумент правых частей).
N - порядок системы
RPOINT - указатель на массив аргументов R(1:N) для вычисления правых частей
ZPOINT - указатель на массив результатов Z(1:N)
X1 - значение независимой переменной, равное X+H;
YPOINT1- указатель на массив результатов Y1(1:N) решения в точке X1;
Алгоритм: предсказание $\_Y_i^{n+1} = Y_i^n + h*Y'^n_i$ и поправка $Y_i^{n+1} = Y_i^n + h/2*(Y'^n_i+\_Y'_i)$.

Описание в PL/1:

```
DCL EILER1STEP ENTRY(FLOAT,PTR,FLOAT,FIXED,
                     ENTRY(FLOAT,FIXED,PTR,PTR),FLOAT,PTR);
```

Пример обращения:

```
CALL EILER1STEP(X,YPOINT,H,N,F,X1,Y1POINT);
```

2.9. Процедура интегрирования системы обыкновенных дифференциальных уравнений методом Кутта-Мерсона с автоматическим выбором шага

KUTTAMERSON(N,X,PTR_Y,EPS,H,FCT,FIRST)

N - число уравнений
X - независимая переменная
PTR_Y - указатель на вектор функций Y(1:N), перед обращением переменным X и Y должны быть присвоены начальные значения
EPS - точность
H - длина интервала интегрирования (от X до X+1)
FCT - идентификатор процедуры, вычисляющей вектор производных F(1:N) по значениям независимой переменной X и вектора функций Y(1:N)

FCT(X,YPTR,FPTR); где YPTR - указатель на Y, FPTR - указатель на F

FIRST - логическая величина, при первом обращении она должна иметь значение '1'B. С целью экономии времени, необходимого для выбора подходящего шага, повторные обращения к процедуре для продолжения интегрирования целесообразно производить, придавая этому параметру значение '0'B.

После выхода из процедуры величины X и Y имеют значения, соответствующие концу интервала интегрирования.

Описание в PL/1:

```
DCL KUTTAMERSON ENTRY(FIXED,FLOAT,PTR,FLOAT,FLOAT,
                      ENTRY(FLOAT,PTR,PTR),BIT(1));
```

Пример обращения:

```
CALL KUTTAMERSON(N,X,PTR_Y,EPS,H,FCT,FIRST);
```

Источник: Журнал "Communication of the ACM". Алгоритм 218.

2.10. Процедуры интегрирования дифференциальных уравнений с автоматическим выбором шага

```
RK2VH(T,YPOINT,H,F,N,EPS,TK)
URKVH(T,YPOINT,H,F,N,M,EPS,TK)
```

Каждая из этих процедур интегрирует систему уравнений вида

```
y1= F1(t, y1,...,yn)
.....
yn=Fn(t, y1,...,yn)
```

на любом заданном отрезке [to,tкон] от начальных значений y1(to),...,yn(to) с переменным (автоматически выбираемым) шагом интегрирования h. Процедура RK2VH использует простой метод второго порядка с погрешностью на шаге O (h**3), а URKVH - более точный метод Рунге-Кутта четвертого порядка с погрешностью на шаге O (h**5). Эти процедуры особенно эффективны при интегрировании сложных систем, имеющих нелинейности в правых частях типа разрывов или изломов, которые происходят в неизвестные заранее моменты времени, а также в тех случаях, когда степень гладкости решения сильно меняется в ходе интегрирования.

T - аргумент системы. Перед обращением к процедуре этому параметру должно быть присвоено его начальное значение. После выхода из процедуры он принимает значение конечного времени, до которого дошел процесс интегрирования. Это может быть либо заданное конечное время tкон ,либо

момент, на котором произошло обнуление шага h из-за задания слишком малого значения EPS;

YPOINT- указатель на массив искомых функций системы Y(1:N). Перед обращением к процедуре в этот массив должны быть помещены начальные значения. После выхода из процедуры получаются результаты интегрирования y1(tк),...,yn(tк);

H - шаг интегрирования. Перед обращением H должно быть присвоено хотя бы грубо ориентировочное начальное значение шага. После выхода H принимает значение последнего не усеченного шага интегрирования. По этому значению можно видеть, какой шаг выбирала машина в конце отрезка интегрирования.

F - идентификатор процедуры, вычисления правых частей.

F(T,YPTR,ZPTR); где

T - время,

YPTR - указатель на массив аргументов правых частей Y,

ZPTR - указатель на массив результатов вычисления функций Z;

N - порядок системы.

EPS - требуемая относительная точность интегрирования на одном шаге. Практически следует брать E-7<EPS<E-1. Чем меньше EPS, тем более мелкие значения шага (в среднем) будет выбирать машина. Если положить EPS=0, то вычисления будут происходить с постоянным шагом, равным заданному начальному.

TK - заданное конечное значение t;

M - параметр, применяемый только в URKVH; Номер функции (1<=M<=N), по которой должен вестись контроль точности интегрирования для выбора шага. Если при обращении к URKVH положить M=0, то контроль будет происходить по той функции, для которой получилась максимальная относительная ошибка (в RK2VH так делается всегда).

H - длина интервала интегрирования (от X до X+1)

Описание в PL/1:

```
DCL RK2VH ENTRY (FLOAT,PTR, FLOAT, ENTRY(FLOAT,PTR,PTR),
                 FIXED,FLOAT,FLOAT);
DCL URKVH ENTRY(FLOAT,PTR, FLOAT, ENTRY(FLOAT,PTR,PTR),
                 FIXED,FIXED,FLOAT,FLOAT);
```

Пример обращения:

```
CALL RK2VH(T,YPOINT,H,F,N,EPS,TK);
CALL URKVH(T,YPOINT,H,F,N,M,EPS,TK);
```

Источник: Ю.К. Любимов и А.А. Газиева "Алгоритмы метода Монте-Карло и численного интегрирования дифференциальных уравнений с автоматическим выбором шага", М., ОТТИ, 1970.

3. РЕШЕНИЕ УРАВНЕНИЙ И СИСТЕМ УРАВНЕНИЙ

3.1. Процедура нахождения корня уравнения

```
ROOT1(A,H0,EPS,K,X0,ADDR(R)) RETURNS(FIXED);
```

Процедура ROOT1 осуществляет поиск корней уравнения на конечном отрезке, если известно наименьшее расстояние между корнями

A(1), A(2) - начало и конец отрезка;

H0 - наименьшее расстояние между корнями (шаг),

EPS - точность вычисления корня, считается, что корень найден, если ABS(F(X)) < EPS;

K - количество корней, которые нужно найти;

X0 - начальная точка из заданного отрезка;

R - массив (1:N), где начиная с R(1), располагаются N вычисленных корней. Значение N присваивается идентификатору ROOT.

Еще должно быть описание в PL/1:

```
DCL PRCROOT ENTRY(FLOAT) EXT VARIABLE RETURNS(FLOAT);
```

Перед обращением к процедуре ROOT параметру PRCROOT необходимо присвоить имя процедуры, вычисляющей значение заданной функции.

Описание в PL/1:

```
DCL ROOT1 ENTRY(FLOAT,FLOAT,FLOAT,FIXED,FLOAT,PTR)
                RETURNS(FIXED);
```

Пример обращения: I=ROOT1(A,H0,EPS,K,X0,ADDR(R));

Источник: Алгоритм ЦКБЭМ

3.2. Процедура нахождения корня уравнения y=f(y)

ROOT2(F,A,EPS,N,G,C,M);

Процедура ROOT2 для нахождения корня уравнения y=f(y)

F - процедура-функция для вычисления f(y);
A - начальное приближение к корню;
EPS - допустимая относительная погрешность в величине корня;
N - максимально допустимое число итераций;
G - искомый корень;
C - невязка f(G)-G;
M - параметр, характеризующий успешность выполнения процедуры. Если отклонение выше допустимого, то M<0. Число ABS(M)-1 показывает, сколько раз поправка превосходила предыдущую.

Если разность f(y)-y имеет одно и то же значение для двух последовательных приближений к корню, то происходит выход из процедуры.

Описание в PL/1:

```
DCL ROOT2 ENTRY(ENTRY(FLOAT) RETURNS(FLOAT) FLOAT,
                FLOAT,FIXED,FLOAT,FLOAT,FIXED);
```

Пример обращения: CALL ROOT2(F,A,EPS,N,G,C,M);

Источник: М.И. Агеев. Алгоритм 26а.

3.3. Процедура нахождения корня уравнения методом деления пополам

BISEC(A,B,EPS,EPS1,F,ERROR,X)

A и B - границы интервала, на котором ищется корень уравнения f(x)=0.
F - имя процедуры-функции; предполагается что в точках A и B она принимает значения разных знаков. Если это условие не выполняется, то происходит возврат в вызывающую программу (ERROR=1).
ERROR - код ошибки, принимает значение 1, если произошел выход по ошибке, в противном случае ERROR=0;
X - полученный корень уравнения;
Корень ищется методом деления отрезка пополам. Итерация прекращается, когда найдена точка X , в которой ABS(F(X))<EPS, или когда длина интервала делается меньше EPS1.

Описание в PL/1:

```
DCL BISEC ENTRY(FLOAT,FLOAT,FLOAT,FLOAT,
                ENTRY(FLOAT) RETURNS(FLOAT),FIXED,FLOAT);
```

Пример обращения: CALL BISEC(A,B,EPS,EPS1,F,ERROR,X);

Источник: М.И. Агеев. Алгоритм 4а.

3.4. Процедура определения всех корней алгебраического многочлена по методу Воеводина

КОЧN(N,BETTA,GAMMA,Q,L,APOINT,BPOINT)

Процедура вычисляет корни многочлена n-ой степени:

$$(A_{2*N} + i*A_{2*N+1})*X^{n} + (A_{2*N-2} + i*A_{2*N-1})*X^{n-1} + \ldots + (A_0 + i*A_1) = 0,$$

где A_J - действительные числа, J=0,1,....2.

N - степень многочлена

APOINT - указатель на A(0:2*N+1) массив коэффициентов заданного многочлена со свободным членом A(0)+I*A(1). Если коэффициенты действительны, то мнимые части нужно задать нулями;

BPOINT- указатель на массив B(1:2*N), в котором задаются начальные приближения и помещаются полученные результаты, т.е. корни многочлена в виде: X_k=B(J)+iB(J+1), J=1,2,...,2N-1; k=(J+1)/2, k=1,2,...N;

BETTA - логическая переменная; если задается '1'B значит необходимо дополнительно уточнять каждый найденный корень для получения большей точности, в противном случае - BETTA='0'B;

GAMMA - логическая переменная; GAMMA='1'B, если коэффициенты многочлена действительные числа (задается для экономии); GAMMA='0'B, если коэффициенты многочлена комплексные числа;

Q - логическая переменная; Q='1'B, если начальные значения корней задаются и Q='0'B, если начальные значения корней не задаются

L - число верных знаков в результате, 0<L<7

Описание в PL/1:

```
DCL КОЧN ENTRY(FIXED,BIT(1),BIT(1),BIT(1),FIXED,PTR,PTR);
```

Пример обращения:

```
CALL КОЧN(N,BETTA,GAMMA,Q,L,APOINT,BPOINT);
```

3.5. Процедура нахождения корней произвольной функции методом Мюллера

```
MULLER: PROCEDURE(F,P1,P2,P3,N,MAXIT,EP1,EP2,REAL,CONJ,
                  ZRPOINT,ZIPOINT,CONVPOINT);
```

Процедура реализует метод Мюллера для нахождения N корней (действительных и комплексных) произвольной комплекснозначной функции f(z).

Каждый корень находится итерациями (квадратичная экстраполяция). При обращении необходимо задать три действительных значения P1,P2,P3 аргумента (практически произвольных) для начала итерационного процесса. После работы процедуры в массивах ZR, ZI(1:N) находятся действительные и мнимые части найденных корней, а в логическом массиве CONV(1:N) - '1'b или '0'b в зависимости от того, сошелся или нет итерационный процесс для соответствующего корня

F - идентификатор процедуры f(x,y,u,v), которая по действительной и мнимой части x и y аргумента Z вычисляет действительную и мнимую часть u и v функции f(Z). При работе процедура F не должна менять значений аргументов x и y;

P1, P2, P3 - начальные значения для итерационного процесса;

N - число отыскиваемых корней;

MAXIT - максимальное число итераций, которое можно делать для вычисления каждого корня;

EP1, EP2 - считается, что корень найден, если модуль разности между двумя последовательными приближениями <EP1 или найдена точка, в которой значение функции <EP2;

REAL - если эта логическая переменная имеет значение '1'b, то ищутся лишь действительные корни; функция f(Z) должна в этом случае принимать только действительные значения (т.е. v=0);

CONJ - если эта логическая переменная имеет значение '1'b , то для каждого найденного корня x, y комплексно-сопряженное с ним число x, -y без дополнительного анализа включается в число корней;

ZRPOINT, ZIPOINT - указатели на массивы ZR и ZI(1:N), в которые процедура помещает действительную и мнимую части найденных корней либо точек, на которых был прекращен процесс (из-за того, что число итераций превысило MAXIT);

CONVPOINT - указатель на логический массив CONV(1:N) в элементы которого процедура помещает '1'b, если считается, что соответствующий корень найден, и '0'b, если итерационный процесс был оборван;

Описание в PL/1:

```
DCL MULLER ENTRY(ENTRY(FLOAT,FLOAT,FLOAT,FLOAT),
                       FLOAT,FLOAT,FLOAT,FIXED,FIXED,
                       FLOAT,FLOAT,BIT(1),BIT(1),
                       PTR,PTR,PTR);
```

Пример обращения:

Для нахождения всех корней уравнения Z**10-1=0 обращение к процедуре может выглядеть так:

```
CALL MULLER(F,0.5,0.6,0.7,10,1000,E-6,E-6,'0'B,'1'B,
                ADDR(ZR),ADDR(ZI),ADDR(CONV));
```

3.6. Процедура решения системы алгебраических уравнений методом Гаусса

```
GAUSS: PROC(PTRA,N1,PTRX,ERROR);
```

PTRA - указатель на расширенную матрицу системы A(1:N1,1:N1+1)

N1 - число неизвестных;

PTRX - указатель на вектор-решение X(1:N1);

ERROR - код ошибки, принимает значение 1, если произошел выход по ошибке, в противном случае ERROR=0;

Описание в PL/1: `DCL GAUSS ENTRY(PTR,FIXED,PTR,FIXED);`

Пример обращения: `CALL GAUSS(ADDR(A),4,ADDR(X),ERROR);`

Источник: Журнал "Communication of the ACM". Алгоритм 126.

3.7. Процедура решения системы алгебраических уравнений методом оптимального исключения

```
SISTEMA: PROC(M,PTRB,KOEF);
```

M - порядок решаемой системы, FIXED;

BPTR - указатель на массив размерности (1:M+1), элементам которого на каждом этапе исключения посредством процедуры KOEF должны быть присвоены значения коэффициентов и свободного члена соответствующего уравнения, после выполнения процедуры в этот же массив переписывается решение системы;

KOEF - идентификатор процедуры, вычисляющей коэффициенты

уравнения KOEF(N,K,BPTR); описание в PL/1: DCL KOEF ENTRY(FIXED,FIXED,PTR);

N - порядок системы,

BPTR - указатель на массив, в который процедура KOEF записывает K-тую строку расширенной матрицы системы.

Процедура предназначена для решения линейных алгебраических уравнений высокого порядка. В отличие от других схем решения метод оптимального исключения не требует одновременного хранения в памяти всей матрицы коэффициентов системы.

При решении системы этим способом на каждом этапе исключения вычисляются коэффициенты только одного очередного уравнения. Преобразование системы производится таким образом, чтобы после каждого этапа коэффициенты при неизвестных X_k, стоящие на главной диагонали, были бы равны единице, а при других k-1 неизвестных X_i, для которых i<k - нулю. Тогда решение системы получается на месте свободных членов после обработки последнего уравнения. Таким образом, в памяти необходимо хранить лишь (n-k)*k коэффициентов преобразованной системы и столбец из k свободных членов.

Описание в PL/1:

```
DCL SISTEMA ENTRY(FIXED,PTR,ENTRY(FIXED,FIXED,PTR));
```

Пример обращения:

```
Z:PROC MAIN;
DCL B(1:3)    FLOAT,
      KOEF   ENTRY(FIXED,FIXED,PTR);

   CALL SISTEMA(2,ADDR(B),KOEF);
   PUT SKIP LIST(B(1),B(2));

   KOEF:PROC(I,J,P);
    DCL (I,J)  FIXED,
          P PTR,
          B(1:3) FLOAT BASED(P);
          IF J=1 THEN DO; B(1)=1; B(2)=3; B(3)=5; RETURN; END;
          IF J=2 THEN DO; B(1)=1; B(2)=9; B(3)=11; END;
           RETURN;
     END KOEF;
END Z;
```

3.8. Процедура решения системы линейных уравнений с ленточной матрицей

BANDSOLVE: PROC(CPOINT,N,M,VPOINT);

CPOINT - указатель на матрицу C(1:N,1:M), которая является ленточной частью матрицы A;

N - порядок матрицы;

M - ширина ленты (обязательно нечетное число);

VPOINT - указатель на массив V(1:N), который вначале содержит вектор B, а после выполнения процедуры BANDSOLVE - решение (вектор X);

Процедура BANDSOLVE находит решение матричного уравнения $\overline{A}\overline{X}=B$.

В матрице A все нулевые элементы содержатся в узкой полоске (ленте), сконцентрированной вдоль главной диагонали. Ленточные элементы i-й строки матрицы A размещаются в i-й строке матрицы C так, что элемент A(i,j) переходит в элемент C(i,j-i+(M+1)/2).

Процедура оперирует только с ленточной частью матрицы A, заданной в массиве C. В массиве C должны присутствовать все ленточные элементы A, как нулевые, так и ненулевые. Значения неопределенных элементов массива C, таких, как C[1,1] или C[N,M], могут быть произвольными.

Преобразование матрицы A в матрицу C производится следующим образом:

```
DO I=1 TO N;
    DO J=1 TO N;
        K=J-I+(M+1)/2; IF 1<=K ! K<=M THEN C(I,K)=A(I,J);
    END;
END;
```

Описание в PL/1: DCL BANDSOLVE ENTRY(PTR,FIXED,FIXED,PTR);

Пример обращения: CALL BANDSOLVE(ADDR(C),12,5,ADDR(V));

Источник: М.И. Агеев. Алгоритм 195a.

3.9. Процедура решения системы линейных уравнений методом сопряженных градиентов

CONGRADIENT: PROC(N,APOINT,XPOINT,RPOINT);

N - порядок системы;
APOINT - указатель на матрицу системы A(1:N,1:N)
XPOINT - указатель на вектор-решение X(1:N), на входе в процедуру вектор X должен содержать начальное приближение;
RPOINT - указатель на вектор правых частей R(1:N);

Процедура решает систему $\overline{A}\overline{X}=R$ методом сопряженных градиентов.

Описание в PL/1: DCL CONGRADIENT ENTRY(FIXED,PTR,PTR,PTR);

Пример обращения:
CALL CONGRADIENT(6,ADDR(A),ADDR(X),ADDR(R));

Источник: М.И. Агеев. Алгоритм 238а.

3.10. Процедура решения системы линейных алгебраических уравнений методом Зейделя

SEIDEL: PROC(N,APOINT,BPOINT,EPS,MAX,ERROR,XPOINT);

N - число неизвестных;
APOINT - указатель на матрицу системы A(1:N,1:N);
BPOINT - указатель на вектор правых частей B(1:N);
EPS - точность решения;
MAX - максимально допустимое число итераций;
ERROR - код ошибки, принимает значение 1, если произошел выход по ошибке, в противном случае ERROR=0;
XPOINT - указатель на вектор-решение X(1:N), на входе в процедуру вектор X должен содержать начальное приближение;

Алгоритм можно применять только для матриц A, не имеющих нулевых диагональных элементов. Если матрица A задается пользователем, то она должна быть предварительно приведена к такому виду.

Описание в PL/1:
DCL SEIDEL ENTRY(FIXED,PTR,PTR,FLOAT,FIXED,FIXED,PTR);

Пример обращения:
```
CALL SEIDEL(N,ADDR(A),ADDR(B),.0001,12,ERROR,ADDR(X));
```

Источник: М.И. Агеев. Алгоритм 220a.

3.11. Процедура-функция для определения корня по методу хорд

```
MXORD: PROC (F,A,B,EPS) RETURNS(FLOAT);
```

F - процедура-функция: `DCL F ENTRY (FLOAT) RETURNS(FLOAT);`
A - начальное значение аргумента;
B - конец отрезка, на котором ищется корень функции F(t) по методу хорд;
EPS - точность;

Описание в PL/1:
```
DCL MXORD ENTRY (ENTRY(FLOAT)
                        RETURNS (FLOAT),FLOAT,FLOAT,FLOAT)
                RETURNS (FLOAT);
```

Пример обращения: `KOR=MXORD(F,A,B,.0001);`

3.12. Процедура решения системы линейных алгебраических уравнений методом вращения

```
MVR: PROC(APOINT,BPOINT,N,M);
```

APOINT - указатель на матрицу системы A(1:N-1,1:N-1)
BPOINT - указатель на матрицу правых частей B(1:N-1,1:M), строками которой являются столбцы различных правых частей;
N - порядок системы;
M - число столбцов правых частей;

Решения получаются на месте правых частей (в соответствующих строках матрицы B). Алгоритм предназначен для одновременного решения нескольких систем линейных алгебраических уравнений с одной и той же матрицей коэффициентов и различными столбцами правых частей.

Описание в PL/1: `DCL MVR ENTRY(PTR,PTR,FIXED,FIXED);`

Пример обращения: `CALL MVR(ADDR(A),ADDR(B),6,3);`

3.13. Процедура решения системы линейных алгебраических уравнений методом ортогонализации

MORT: PROC(APOINT,N,PI2);

N - порядок системы;

APOINT - указатель на матрицу A порядка N+1, строками которой являются строки расширенной матрицы системы с добавлением строки (0,0,...,0,1);

PI2 - признак, дающий возможность получать решение с большей (PI2='1'b) или меньшей (PI2='0'b) точностью. Решение получается на месте первых N элементов N+1-й строки матрицы A.

Описание в PL/1: DCL MORT ENTRY(PTR,FIXED,BIT(1));

Пример обращения: CALL MORT(ADDR(A),4,'1'b);

4. ОПЕРАЦИИ С МАТРИЦАМИ

4.1. Процедура перемножения матриц, расположенных в одном одномерном массиве

ММЗ: PROC(П,Q,R,APOINT,I,J,Д);

П - количество столбцов первой матрицы и количество строк второй матрицы;

Q - количество строк первой матрицы;

R - количество столбцов второй матрицы;

APOINT - указатель на массив A(1:К), в котором помещаются матрицы и результат (К>=П*Q+П*R+Q*R);

I - целое число, обозначающее количество элементов массива, стоящих впереди результата перемножения матриц;

J - целое число, обозначающее количество элементов массива, стоящих впереди первой матрицы;

Д - целое число, обозначающее количество элементов массива, стоящих впереди второй матрицы;

Матрицы записываются строка за строкой.

Описание в PL/1:

```
DCL ММЗ ENTRY(FIXED,FIXED,FIXED,PTR,FIXED,FIXED,FIXED);
```

Пример обращения:

```
CALL ММЗ(10,6,5,ADDR(A),I,0,10*6);
```

4.2. Процедура поиска максимального числа в матрице и запоминания его координат

МАКС1: PROC(XPOINT,П,Т,У,I,J);

XPOINT - указатель на исходную матрицу X(1:П,1:Т);

П - количество строк в матрице;

Т - количество столбцов в матрице;

У - результат (максимум);

I - номер строки, в которой находится максимальное число;

J - номер столбца, в котором находится максимальное число;

Описание в PL/1:

```
DCL МАКС1 ENTRY(PTR,FIXED,FIXED,FLOAT,FIXED,FIXED);
```

Пример обращения:

```
CALL МАКС1(ADDR(A),16,10,YMAX,I,J);
```

4.3. Процедура поиска минимального числа в матрице и запоминания его координат

МИН1: PROC(XPOINT,П,Т,У,I,J);

XPOINT - указатель на исходную матрицу X(1:П,1:Т);
П - количество строк в матрице;
Т - количество столбцов в матрице;
У - результат (минимум);
I - номер строки, в которой находится минимальное число;
J - номер столбца, в котором находится минимальное число;

Описание в PL/1:
DCL МИН1 ENTRY(PTR,FIXED,FIXED,FLOAT,FIXED,FIXED);

Пример обращения: CALL МИН1(ADDR(A),16,10,YMIN,I,J);

4.4. Процедура вычисления собственных векторов симметрической матрицы методом Якоби

JACOBI: PROC(APOINT,SPOINT,N,EPS);

N - порядок матрицы;
APOINT - указатель на матрицу A(1:N,1:N), для которой ищутся собственные векторы и значения. После работы процедуры элемент матрицы A(к,к) равен соответствующему собственному значению;
SPOINT - указатель на матрицу S(1:N,1:N). После работы процедуры к-я строка матрицы S содержит к-й собственный вектор;
EPS - число, обуславливающее точность процесса: итерации прекращаются, когда все недиагональные элементы делаются меньше, чем

$$EPS*SQRT(\sum_{i\sim=s.} A_{i,s}^2)/N,$$ (где i~=s).

Описание в PL/1: DCL JACOBI ENTRY(PTR,PTR,FIXED,FLOAT);

Пример обращения: CALL JACOBI(ADDR(A),ADDR(X),12,.0001);

Источник: Журнал "Communication of the ACM", т.5, #4 (1962). Стр.208 алгоритм 85.

4.5. Процедура обращения матрицы по методу Гаусса

INVERT: PROC(APOINT,N,EPS,DET,ALARM);

N - порядок матрицы;
APOINT - указатель на обращаемую матрицу A(1:N,1:N). Обратная матрица будет помещена на место матрицы A.
DET - переменная, значение которой после работы процедуры будет равно определителю матрицы A (если не произойдет аварийного выхода).
ALARM - код ошибки, принимает значение 1, если произошел выход по ошибке (когда в процессе обращения матрицы A какой-либо из ее главных (диагональных) элементов окажется меньше EPS), в противном случае ALARM=0;

Обращение матрицы производится по методу Гаусса без выбора максимального элемента. Процедура очень простая и короткая, но при обращении плохо обусловленных матриц могут возникать большие ошибки.

Описание в PL/1:
DCL INVERT ENTRY(PTR,FIXED,FLOAT,FLOAT,FIXED);

Пример обращения: CALL INVERT(ADDR(A),6,.0001,DET,ALARM);

Источник: Журнал "Communication of the ACM". Алгоритм 140.

4.6. Процедура обращения матрицы по методу Гаусса-Жордана

INVERT1: PROC(APOINT,N,EPS,DET,ALARM);

N - порядок матрицы;
APOINT - указатель на обращаемую матрицу A(1:N,1:N). Обратная матрица будет помещена на место матрицы A.
DET - переменная, значение которой после работы процедуры будет равно определителю матрицы A (если не произойдет аварийного выхода).
ALARM - код ошибки, принимает значение 1, если произошел выход по ошибке (когда в процессе обращения матрицы A какой-либо из ее главных (диагональных) элементов окажется меньше EPS), в противном случае ALARM=0;

Обращение матрицы производится по методу Гаусса-Жордана с выбором максимального элемента в строке.

Описание в PL/1:
DCL INVERT1 ENTRY(PTR,FIXED,FLOAT,FLOAT,FIXED);

Пример обращения: CALL INVERT1(ADDR(A),6,.0001,DET,ALARM);

Источник: Журнал "Communication of the ACM". Алгоритм 120.

4.7. Процедура обращения матрицы по методу Гаусса-Жордана с максимальным не преобразованным элементом

INVERT2: PROC(APOINT,N,EPS,ALARM);

N - порядок матрицы;
APOINT - указатель на обращаемую матрицу A(1:N,1:N). Обратная матрица будет помещена на место матрицы A.
ALARM - код ошибки, принимает значение 1, если произошел выход по ошибке (когда в процессе обращения матрицы A какой-либо из ее главных (диагональных) элементов окажется меньше EPS), в противном случае ALARM=0;

Обращение матрицы производится по методу Гаусса-Жордана с выбором на каждом шаге в качестве главного элемента максимального элемента в еще не преобразованной части матрицы.

Описание в PL/1: DCL INVERT2 ENTRY(PTR,FIXED,FLOAT,FIXED);

Пример обращения: CALL INVERT2(ADDR(A),6,.0001,ALARM);

Источник: Журнал "Communication of the ACM". Алгоритмы 230 и 231.

4.8. Процедура обращения матрицы методом жордановских исключений

БИНВИ: PROC(APOINT,П,BPOINT,H1,H,error);

П - порядок матрицы;
APOINT - указатель на исходную матрицу A(1:П,1:П);
BPOINT - указатель на обратную матрицу B(1:П,1:П) (результат);
H1 – величина $|E-A*Do|$, где Do - обратная матрица;
H2 - величина $|E-A*Dk|$, где Dk - уточненная обратная матрица;
error - код ошибки, принимает значение 1, если произошел выход по ошибке (когда обратная матрица для исходной не существует), в противном случае error=0;

Процедура производит обращение матрицы методом жордановых исключений с глобальным выбором наибольшего по модулю разрешаемого элемента на каждом шаге исключений. Перед обращением матрицы производится выравнивание порядков по строкам. После обращения и демасштабирования процедура производит итерационное уточнение элементов обратной матрицы по методу Фадеева. Цикл уточнений заканчивается в момент прекращения монотонного убывания величины:

|E-A*Dk|, где E-единичная матрица; A-исходная матрица; Dk-уточненная обратная матрица на k-м шаге.

Описание в PL/1:

```
DCL БИНВИ ENTRY(PTR,FIXED,PTR,FLOAT,FLOAT,FIXED);
```

Пример обращения:

```
CALL БИНВИ(ADDR(A),10,ADDR(X),H1,H2,ERROR);
```

4.9. Процедура-функция для вычисления определителя

```
DETERMINANT: PROC(APOINT,N) RETURNS(FLOAT);
```

N - порядок матрицы;
APOINT - указатель на матрицу A(1:N,1:N)

Определитель вычисляется путем приведения матрицы A к треугольному виду с выбором максимального элемента в каждой строке. Перед приведением строки матрицы A нормируются. В процессе работы процедуры матрица A портится.

Описание в PL/1:

```
DCL DETERMINANT ENTRY(PTR,FIXED) RETURNS(FLOAT);
```

Пример обращения: `D1=DETERMINANT(ADDR(A),12);`

Источник: Журнал "Communication of the ACM". Алгоритм 269.

4.10. Процедура вычисления определителя комплексной матрицы

DET: PROCEDURE(REALPOINT,IMGNPOINT,N,RED,IMD,ERROR);

REALPOINT - указатель на REAL(1:N,1:N) - действительную часть матрицы;
IMGNPOINT - указатель на IMGN(1:N,1:N) - мнимую часть матрицы;
N - порядок матрицы;
RED - действительная часть полученного определителя;
IMD - мнимая часть полученного определителя;
ERROR - код ошибки. В случае нормального завершения процедуры равен 0. В случае, если абсолютная величина действительной или мнимой части определителя данной матрицы достигнет 1.E19 произойдет выход по ошибке, ERROR тогда будет равен 1.

Описание в PL/1:
DCL DET ENTRY(PTR,PTR,FIXED,FLOAT,FLOAT,FIXED);

Пример обращения: CALL DET(ADDR(AR),ADDR(AI),6,DR,DI,ERROR);

4.11. Процедура транспонирования прямоугольной матрицы

TPAH: PROC(APOINT,BPOINT,M,N);

APOINT - указатель на исходную матрицу A(1:M,1:N);
BPOINT - указатель на транспонированную матрицу B(1:N,1:M);
N - число строк исходной матрицы;
M - число столбцов исходной матрицы;

Описание в PL/1: DCL TPAH ENTRY (PTR,PTR,FIXED,FIXED);

Пример обращения: CALL TPAH(ADDR(A),ADDR(B),8,10);
Обращение вида CALL TPAH(ADDR(A),ADDR(A),8,10); не допускается.

4.12. Процедура перемножения матриц

УММ: PROC(APOINT,BPOINT,P,CPOINT,M,Q);

APOINT - указатель на матрицу A(1:M,1:P);
BPOINT - указатель на матрицу B(1:P,1:Q);
CPOINT - указатель на матрицу-результат C(1:M,1:Q);
M - число строк матриц A и C;
P - число столбцов матрицы A и число строк матрицы B;
Q - число столбцов матриц B и C;

Описание в PL/1:
DCL УММ ENTRY(PTR,PTR,FIXED,PTR,FIXED,FIXED);

Пример обращения:
CALL УММ(ADDR(A),ADDR(B),PP,ADDR(C),MM,QQ);

Обращение вида
CALL УММ(ADDR(A),ADDR(B),PP,ADDR(A),MM,QQ); и
CALL УММ(ADDR(A),ADDR(B),PP,ADDR(B),MM,QQ); не допускается.

4.13. Процедура умножения матрицы на вектор

УМВ: PROC(APOINT,BPOINT,M,CPOINT,N);

APOINT - указатель на матрицу A(1:M,1:N);
BPOINT - указатель на вектор B(1:N);
CPOINT - указатель на вектор-результат C(1:M);
M - число строк матрицы A;
N - число столбцов матрицы A;

Описание в PL/1: DCL УМВ ENTRY(PTR,PTR,FIXED,PTR,FIXED);

Пример обращения: CALL УМВ(ADDR(A),ADDR(B),M,ADDR(C),N);

Обращение вида
CALL УМВ(ADDR(A),ADDR(B),M,ADDR(B),N); не допускается.

4.14. Процедура транспонирования квадратной матрицы

TPAK: PROC(N,APOINT);

APOINT - указатель на матрицу A(1:N,1:N);
N - порядок матрицы A;

Описание в PL/1: DCL TPAK ENTRY(FIXED,PTR);

Пример обращения: CALL TPAK(6,ADDR(C));

4.15. Процедура умножения матрицы на диагональную матрицу

YMD: PROC(APOINT,BPOINT,M,CPOINT,N);

APOINT - указатель на матрицу A(1:M,1:N);
BPOINT - указатель на вектор B(1:N), соответствующий диагональной матрице;
CPOINT - указатель на матрицу результатов C(1:M,1:N);
M - число строк матрицы A;
N - число столбцов матрицы A;

Описание в PL/1: DCL YMD ENTRY(PTR,PTR,FIXED,PTR,FIXED);

Пример обращения: CALL YMD(ADDR(A),ADDR(B),M,ADDR(C),N);

4.16. Процедура умножения диагональной матрицы на матрицу

YDM: PROC(APOINT,BPOINT,M,CPOINT,N);

APOINT - указатель на матрицу A(1:M,1:N);
BPOINT - указатель на вектор B(1:M), соответствующий диагональной матрице;
CPOINT - указатель на матрицу результатов C(1:M,1:N);
M - число строк матрицы A и C;
N - число столбцов матрицы A и C;

Описание в PL/1: DCL YDM ENTRY(PTR,PTR,FIXED,PTR,FIXED);

Пример обращения: CALL YDM(ADDR(A),ADDR(B),M,ADDR(C),N);

4.17. Процедура выравнивания элементов матрицы

VIR: PROC(APOINT,BPOINT,BETA1POINT,BETA2POINT,N);

APOINT - указатель на исходную матрицу A(1:N,1:N);
BPOINT - указатель на результат-матрицу B(1:N,1:N);
BETA1POINT - указатель на вектор BETA1(1:N);
BETA2POINT - указатель на вектор BETA2(1:N);
N - порядок матриц A и B;

Процедура VIR применяется обычно для последующего обращения выровненной матрицы. Между матрицами A и B существуют соотношения:

$$B = B1 * A * B2,$$
$$A^{-1} = B2^{-1} * B * B1,$$

где B1, B2 - диагональные матрицы порядка N, чьи диагональные элементы получаются соответственно в массивах BETA1 и BETA2.

Описание в PL/1: DCL VIR ENTRY(PTR,PTR,PTR,PTR,FIXED);

Пример обращения:
CALL VIR(ADDR(A),ADDR(B),ADDR(BETA1),ADDR(BETA2),N);

5. ИНТЕРПОЛЯЦИЯ, АППРОКСИМАЦИЯ, СГЛАЖИВАНИЕ

5.1. Процедура линейной интерполяции функции одной переменной

```
LININTERP: PROC(XPOINT,УPOINT,XX,УУ);
```

XPOINT - указатель на массив аргументов X(0:N);
УPOINT - указатель на массив соответствующих значений функции У(0:N);
Линейной интерполяцией находится значение УУ, соответствующее значению XX, удовлетворяющему условию X(0) <= XX < X(N);

Описание в PL/1: `DCL LINITERP ENTRY(PTR,PTR,FLOAT,FLOAT);`

Пример обращения: `CALL LININTERP(ADDR(X),ADDR(У),XX,УУ);`

5.2. Процедура-функция интерполирования функции одной переменной, заданной таблицей с переменным шагом, методом Лагранжа

```
ЛАГРАНЖ1: PROC(XPOINT,УPOINT,X1,P,П) RETURNS(FLOAT);
```

XPOINT - указатель на массив значений переменной X(1:П);
УPOINT - указатель на массив значений интерполируемой функции У(1:П);
X1 - значение аргумента;
P - степень интерполяционного полинома Лагранжа;
П - размерность массивов;

Процедура ЛАГРАНЖ1 вычисляет значение полинома Лагранжа L(X) степени P в точке X1 (P<П). Значения аргумента в массиве X должны быть упорядочены в порядке возрастания. Если значение X1 выходит за границы диапазона, заданного массивом X, то процедура возвращает 0.

Описание в PL/1:

```
DCL ЛАГРАНЖ1 ENTRY (PTR, PTR, FLOAT, FIXED, FIXED)
                   RETURNS(FLOAT);
```

Пример обращения: `У1=ЛАГРАНЖ1(ADDR(X),ADDR(У),X1,2,N);`

5.3. Процедура-функция интерполирования функции двух переменных, заданных таблицами с переменным шагом, методом Лагранжа

```
ЛАГРАНЖ2: PROC (APOINT, XPOINT ,УPOINT ,X1 ,X2 ,P ,П ,П1)
RETURNS(FLOAT);
```

APOINT - указатель на массив значений первой переменной A(1:П1);
XPOINT - указатель на массив значений второй переменной X(1:П);
УPOINT - указатель на массив значений интерполируемой функции У(1:П1*П);
X1 - значение первой переменной;
X2 - значение второй переменной;
P - степень интерполяционного полинома Лагранжа;
П, П1 - размерность массивов;

Процедура ЛАГРАНЖ2 вычисляет значение полинома Лагранжа L(A,X) степени P в точке (X1,X2) (P<П,P<П1). Значения аргументов в массивах X и A должны быть упорядочены в порядке возрастания. Если значения X1 и X2 выходят за границы диапазонов, заданных массивами X и A, то процедура возвращает 0.

Описание в PL/1:

```
DCL ЛАГРАНЖ2
ENTRY (PTR, PTR, PTR, FLOAT, FLOAT, FIXED, FIXED, FIXED)
RETURNS(FLOAT);
```

Пример обращения:

```
Y1=ЛАГРАНЖ2(ADDR(A),ADDR(X),ADDR(Y),X1,X2,4,N1,N2);
```

5.4. Процедура квадратичной интерполяции функций, заданных таблично с равномерным шагом

```
PARINT: PROC(УPOINT,У0POINT,X,X0,H1,C,N);
```

УPOINT - указатель на массив результатов интерполяции У(1:N), т.е. значений функций для заданного аргумента X;
У0POINT - указатель на заданный массив значений функций в узлах интерполяции У0(1:N,0:C);
X - аргумент;
X0 - начальное значение аргумента;
H1 - шаг таблицы функций $H1=X_{i+1}-X_i$,i=0,1,2...;

C - число узлов интерполяции, уменьшенное на единицу (количество интервалов);

N - количество интерполируемых функций;

Процедура PARINT позволяет интерполировать одновременно N функций одного и того же аргумента, заданных с одним и тем же равномерным шагом и одним началом отсчета. Рекомендуется использовать процедуру на монотонных участках, в противном случае интерполирование не всегда дает удовлетворительный результат.

Описание в PL/1:

```
DCL PARINT ENTRY(PTR,PTR,FLOAT,FLOAT,FLOAT,FIXED,FIXED);
```

Пример обращения: `CALL PARINT(ADDR(У),ADDR(У0),X,X0,H1,C,N);`

5.5. Процедура интерполяции по узлам методом Лагранжа

```
LAGRANGE: PROC(N,U,XPOINT,УPOINT,ANS);
```

N - степень полинома L(X);

U - аргумент интерполяции;

XPOINT - указатель на массив абцисс узлов интерполяции X(1:N+1);

УPOINT - указатель на массив ординат У(1:N+1);интерполируемой функции f

ANS - результат: L(U)~=f(U);

Процедура LAGRANGE вычисляет полином Лагранжа L(X) N-й степени (по N+1 данным узлам интерполяции) по данному значению U, при котором требуется интерполяция.

Описание в PL/1:

```
DCL LAGRANGE ENTRY(FIXED,FLOAT,PTR,PTR,FLOAT);
```

Пример обращения: `CALL LAGRANGE(3,U,ADDR(X),ADDR(У),L);`

5.6. Процедура интерполяции по Эрмиту

HERMITE: PROC(N,U,XPOINT,УPOINT,У1POINT,ANS);

N - степень полинома H(X);
U - аргумент интерполяции;
XPOINT - указатель на массив X(1:N+1) абсцисс узлов интерполяции;
УPOINT - указатель на массив ординат У(1:N+1) интерполируемой функции f;
У1POINT - указатель на массив У1(1:N+1) значений производной функции f
ANS - результат: H(U)~=f(U);

Процедура HERMITE вычисляет значение полинома Эрмита H(X) степени 2*N+1 по данным значениям функции и ее производной в каждой из N+1 точек.

Описание в PL/1:
DCL HERMITE ENTRY(FIXED,FLOAT,PTR,PTR,PTR,FLOAT);

Пример обращения:
CALL HERMITE(16,U,ADDR(X),ADDR(У),ADDR(У1),H);

5.7. Процедура сглаживания по трем точкам

SMOOTH13: PROC(N,XPOINT,ERROR);

N - число значений функции,записанных в массиве X;
XPOINT - указатель на массив X(1:N);
ERROR - код ошибки. Если обращение к процедуре производится с N < 3, то происходит аварийный выход из процедуры, ERROR в этом случае равна 1, в противном случае - 0.

Процедура SMOOTH13 использует трехточечные формулы Грама первой степени. Результаты помещаются в массиве X.

Описание в PL/1: DCL SMOOTH13 ENTRY(FIXED,PTR,FIXED);

Пример обращения: CALL SMOOTH13(N,ADDR(X(5)),ERROR);

Источник: М.И.Агеев. Алгоритм 188a.

5.8. Процедура сглаживания по пяти точкам

SMOOTH35: PROC(N,XPOINT,ERROR);

N - число значений функции, записанных в массиве X;
XPOINT - указатель на массив X(1:N);
ERROR - код ошибки. Если обращение к процедуре производится с N < 5, то происходит аварийный выход из процедуры, ERROR в этом случае равна 1, в противном случае - 0.

Процедура SMOOTH35 использует пятиточечные формулы Грама третьей степени. Результаты помещаются в массиве X.

Описание в PL/1: DCL SMOOTH35 ENTRY(FIXED,PTR,FIXED);

Пример обращения: CALL SMOOTH35(N,ADDR(X(5)),ERROR);

Источник: М.И.Агеев. Алгоритм 189а.

5.9. Процедура расчета коэффициентов интерполяции функции f(x) полиномом

ИНТКОЕФ2: PROC(XPOINT,FPOINT,Nx,Ny,П,Т,XX,YY,З,Z,APOINT);

XPOINT - указатель на таблицу аргументов X(0:Nx);
FPOINT - указатель на двумерную таблицу функции F(0:Ny,0:Nx);
Nx - длина таблицы X;
Ny - длина таблицы F;
П - порядок полинома интерполяции по аргументу X;
Т - порядок полинома интерполяции по аргументу У;
XX - значение аргумента X, для которого требуется построить интерполяционный полином.
YY - значение аргумента Y, для которого требуется построить интерполяционный полином.
З - индекс первого из П+1 интерполяционных узлов, по которым будут строиться интерполяционные полиномы от X, определяется в процессе работы процедуры.
Z - индекс первого из Т+1 интерполяционных узлов, по которым будут строиться интерполяционные полиномы от Y, определяется в процессе работы процедуры.
APOINT - указатель на двумерный массив коэффициентов интерполяции

A(0:T,0:П);

Процедура ИНТКОЕФ2 вычисляет коэффициенты каждого из T+1 полиномов от X, построенных по узлам заданной функции F(X,Y), окружающим точку (XX,YY). Необходимо выполнение условий:

X(0) <= XX <= X(Nx)
F(0,З) <= YY <= F(Ny,З)

Процедура ИНТКОЕФ2 использует процедуру ИНТКОЕФ1.

Описание в PL/1:

```
DCL
ИНТКОЕФ2 ENTRY (PTR, PTR, FIXED, FIXED, FIXED,
                FIXED, FLOAT,FLOAT, FIXED,FIXED,PTR);
```

Пример обращения:

```
ИНТКОЕФ2 (ADDR(X),ADDR(Y),Nx,Ny,П,T,XX,YY,З,Z,ADDR(A));
```

5.10. Процедура расчета интерполированного значения

```
ИНТЕРП2: PROC(APOINT,УPOINT,XX,УУ,П,T,З,FF);
```

APOINT - указатель на двумерный массив коэффициентов интерполяции A(0:T,0:П), рассчитанных процедурой ИНТКОЕФ2;

УPOINT - указатель на массив аргументов переменной УУ(0:Ny);

Nx - длина таблицы X;

Ny - длина таблицы F;

П - порядок полинома интерполяции по аргументу X;

T - порядок полинома интерполяции по аргументу У;

XX - значение аргумента X, для которого требуется построить интерполяционный полином.

УУ - значение аргумента У, для которого требуется построить интерполяционный полином.

З - индекс начального узла интерполяции по y, вычисляется процедурой ИНТКОЕФ2 (параметр Z);

FF - интерполированное значение f(x);

Процедура ИНТЕРП2 вычисляет значение f(x) в текущей точке (XX,УУ) по полиномам, коэффициенты которых рассчитаны процедурой ИНТКОЕФ2.

Описание в PL/1:

```
DCL ИНТЕРП2 ENTRY(PTR, PTR, FLOAT, FLOAT, FIXED,
                  FIXED, FIXED,FLOAT);
```

Пример обращения:

```
CALL ИНТЕРП2(ADDR(A),ADDR(Y),XX,YY,П,T,З,FF);
```

5.11. Процедура вычисления по схеме Горнера функции от одного аргумента, заданной полиномами степени t

```
ГОРН: PROC(ВИ,ЮPOINT,ВЖ, Ж30,П,T,P1,ERROR);
```

ВИ - текущее значение аргумента;

ЮPOINT - указатель на массив исходных данных Ю(0:P), имеющий вид:

$Ю=(ВИ_0,А_{10},А_{20},А_{30},...,А_{T+10},ВИ_1,А_{11},А_{21},...ВИ_П)$,

Здесь ВИ(0),ВИ(1),...,ВИ(П-1) - нижние границы отрезков определения полиномов; ВИ(П) - верхняя граница последнего отрезка; A(I,J) - коэффициенты полинома;

ВЖ - индекс элемента массива Ю, начиная с которого располагаются исходные данные, можно, например, полагать ВЖ=1+К*(T+2), 0<=К<=П-2;

Ж30 - если равен 0 - ВИ берется в нормальном виде; 1 - ВИ=ВИ-Ю(К);

П - количество полиномов в исходных данных;

T - степень полиномов;

P1 - результат;

ERROR - код ошибки. При нарушении условия ВИ(0)<=ВИ<=ВИ(П) происходит аварийный выход из процедуры, ERROR в этом случае равна 1, в противном случае - 0.

Процедура ГОРН вычисляет функцию, представленную П полиномами степени T на П участках изменения аргумента.

Описание в PL/1:

```
DCL ГОРН2 ENTRY (FLOAT, PTR, FIXED, FIXED, FIXED,
                 FIXED,FLOAT,FIXED);
```

Пример обращения:

```
CALL ГОРН(X,ADDR(A),N,1,П,T,FF,ERROR);
```

5.12. Процедура вычисления коэффициентов полиномов Чебышева

TCHEBCO: PROC(N,APOINT);

N - степень полинома Чебышева;
APOINT - указатель на массив результатов A;

Процедура TCHEBCO вычисляет коэффициенты полинома Чебышева. Ненулевые коэффициенты запоминаются в массиве A(1:p) в порядке, заданном формулами:

p=(N+2)/2;

$$T_{2*p}(X)=\sum_{i=0}^{p} A(i+1)*X^{**}(2*i);$$ для N - четного

$$T_{2*p+i}(X)=\sum_{i=0}^{p} A(i+1)*X^{**}(2*i+1);$$ для N - нечетного

Описание в PL/1: DCL TCHEBCO ENTRY(FIXED,PTR);

Пример обращения: CALL TCHEBCO(N,ADDR(X));

Источник: М.И.Агеев. Алгоритм 227а.

5.13. Процедура вычисления коэффициентов интерполяционного многочлена Ньютона

NEWINT: PROC(XPOINT,УPOINT,APOINT,N);

XPOINT - указатель на массив узлов интерполяции X(0:N);
УPOINT - указатель на массив ординат У(0:N);
APOINT - указатель на массив искомых коэффициентов A(0:N);
N - степень интерполяционного многочлена.

Процедура NEWINT по значениям функции y0,...,yn, таблично заданной в точках x1,...xn, вычисляет коэффициенты интерполяционного многочлена Ньютона. Результатом является массив коэффициентов a0,...,an интерполяционного полинома Ньютона: $A0+A1*X+...+An*X^{n}$;

Описание в PL/1: DCL NEWINT ENTRY(PTR,PTR,PTR,FIXED);
Пример обращения: CALL NEWINT(ADDR(X),ADDR(У),ADDR(A),8);

5.14. Процедура рациональной интерполяции с помощью непрерывных дробей

```
CONFR: PROC(M,XPOINT,УPOINT,APOINT,BPOINT,DPOINT);
```

M - число точек;
XPOINT - указатель на массив X(1:M);
УPOINT - указатель на массив У(1:M);
APOINT,BPOINT,DPOINT - указатели на массивы коэффициентов A(1:M), B(0:M/2), D(0:M/2) соответствующих дробей.

Процедура CONFR данным m точкам (Xi,Yi) ставит в соответствие непрерывную дробь в форме:

```
                X-X(1)
          A1 + --------
               A2+X-X(2)
                 ------
                 A3+...X-X(m-1)
                      ---------
                        Am
```

Процедура приводит также эту непрерывную дробь к рациональной функции

```
B0+B1*X+...+Bk*X**k
------------------- ,
D0+D1*X+...+Dl*X**L
```

где k и l не больше, чем m/2.

Описание в PL/1:

```
DCL CONFR ENTRY(FIXED,PTR,PTR,PTR,PTR,PTR);
```

Пример обращения:

```
CALL CONFR(16,ADDR(X),ADDR(У),ADDR(A),ADDR(B),ADDR(D);
```

Источник: М.И.Агеев. Алгоритм 18а.

5.15. Процедура нахождения чебышевского приближения

CHEBFIT: PROC(XPOINT,УPOINT,N,APOINT,M);

XPOINT - указатель на массив узлов интерполяции X(1:N);
УPOINT - указатель на массив ординат У(1:N);
N - размермерность X и У.
APOINT - указатель на массив искомых коэффициентов A(0:M+1);
M - степень интерполяционного многочлена.

Процедура CHEBFIT находит многочлен степени M, дающий наилучшее абсолютное приближение для функции У(1:N), заданной в N равноотстоящих точках X(1:N), где N>=M+2. После работы процедуры коэффициенты многочлена будут находится в массиве A(0:M+1). Элемент A(M+1) этого массива будет равен абсолютной величине отклонения многочлена от функции.

Итерационный процесс нахождения многочлена должен заканчиваться за конечное число шагов, однако из-за ошибок округления он иногда может зацикливаться. В этом случае процедура выдаст в массив A соответственно коэффициенты последнего полученного многочлена и со знаком минус абсолютную величину его отклонения.

Описание в PL/1: DCL CHEBFIT ENTRY(PTR,PTR,FIXED,PTR,FIXED);

Пример обращения: CALL CHEBFIT(ADDR(X),ADDR(У),N,ADDR(A),8);

Источник: "Communication of the ACM". Алгоритм 318.

5.16. Процедура среднеквадратичной аппроксимации экспериментальных данных полиномом

POLYFIT: PROC(N0,XPOINT,УPOINT,DУPOINT,L,M,P);

N0 - число точек измерения;
XPOINT - указатель на массив значений независимой переменной X(0:N0-1), (необязательно равноотстоящие);
УPOINT - указатель на массив значений измеряемой величины У(0:N0-1);
DУPOINT - указатель на массив значений среднеквадратичных ошибок DУ(0:N0-1);
L - логическая переменная, которую нужно полагать '1'b, если заданы абсолютные ошибки, и '0'b, если относительные;

M - (<=20) степень аппроксимирующего полинома;

P - одномерный массив (0:20) коэффициентов аппроксимирующего полинома, начиная со свободного члена;

Процедура POLYFIT находит полином степени M<=20, его коэффициенты будут находиться в массиве P.

Описание в PL/1:

```
DCL POLYFIT ENTRY(FIXED,PTR,PTR,PTR,BIT(1),
                  FIXED,(21) FLOAT);
```

Пример обращения:

```
CALL POLYFIT(90,ADDR(X),ADDR(У),ADDR(DУ),'0'b,M,P);
```

Источник: G.E."Generation and use of ortogonal polinomials for data fitting with a digital computer". J.Siam, 1957,5,2,74.

5.17. Процедура построения аппроксимирующего полинома по методу наименьших квадратов для функции, заданной с равномерным и неравномерным шагом

```
МНК: PROC(APOINT,PPOINT,M0,N,MK,H,ERROR);
```

N - количество значений функции yi;

APOINT - указатель на массив A(1:2*N), содержащий значения y1,...,yn,x1,...,xn (в случае когда H=0, тогда таблица значений yi задана неравномерным шагом xi), или A(1:N+1), содержащий значения y1,...,yn,x1 (в этом случае H не должно быть равно 0 т.е. таблица значений yi задана равномерным шагом H);

PPOINT - указатель на массив значений полинома начиная со свободного члена P;

M0,MK - соответственно минимальная и максимальная степень аппроксимирующего полинома;

H - шаг таблицы значений функции yi;

ERROR - код ошибки. В случае ошибки при решении системы алгебраических уравнений происходит аварийный выход из процедуры, ERROR в этом случае равна 1, в противном случае - 0.

Процедура МНК использует процедуру GAUSS.

Описание в PL/1:

```
DCL МНК ENTRY(PTR,PTR,FIXED,FIXED,FIXED,FLOAT,FIXED);
```

Пример обращения:

```
CALL MHK(ADDR(A),ADDR(P),M0,N,MK,H,ERROR);
```

5.18. Процедура аппроксимации поверхности по методу наименьших квадратов

```
SURFIT: PROC(FPOINT,ZPOINT,WPOINT,M,N,APOINT,EPOINT,R,L);
```

M - число точек;

FPOINT - указатель на массив F(1:M,1:N), здесь находятся значения базисных функций f1,f2,...fn, заданных таблично в M точках;

ZPOINT - указатель на массив значений ординат Z(1:M);

WPOINT - указатель на массив весов данных точек W(1:M);

M - число узлов, узлы одни и те же для всех базисных функций и аппроксимируемой функции;

N - число базисных функций;

APOINT - указатель на массив искомых коэффициентов A(1:N);

EPOINT - указатели на массив погрешностей E(1:M);

$$Ei = Zi - \sum_{k=1}^{N} Ak*Fik ;$$

R - средняя длина вектора разностей; R = SQRT(SUM(Ei**2)/M);

L - код ошибки. В случае ошибки при обращении матрицы происходит аварийный выход из процедуры, ERROR в этом случае равна 1, в противном тслучае - 0.

Коэффициенты Ai определяются в процедуре минимизацией выражения

$$\sum_{i=1}^{M} Wi*Ei^2$$

Процедура SURFIT использует процедуру INVERT2

Описание в PL/1:

```
DCL SURFIT ENTRY (PTR, PTR, PTR, FIXED, FIXED,
                  PTR,PTR,FLOAT,FIXED);
```

Пример обращения:

```
CALL SURFIT (ADDR(F),ADDR(Z),ADDR(W), M,N,
             ADDR(A),ADDR(E),R,ERROR);
```

Источник: М.И.Агеев. Алгоритм 176а.

5.19. Процедура аппроксимации функции двух независимых переменных ортогональными полиномами по методу наименьших квадратов

```
SURFACEFIT:
PROCEDURE(XPOINT,UPOINT,УPOINT,WPOINT,ZPOINT,
                NMAX,MMAX,IMAX,JMAX,
                BETAPOINT,PHIPOINT,ZCOMPPOINT,
                INSGD,MINSGDCOMP,SUMDIFCOMP,MAXDIFCOMP,
                MEANX,MEANУ,MEANZ);
```

Входные параметры:

XPOINT - указатель на массив X(1:IMAX) - табличные значения первого аргумента;

UPOINT - указатель на массив U(1:IMAX) - веса для аргумента X;

УPOINT - указатель на массив У(1:JMAX) - табличные значения второго аргумента;

WPOINT - указатель на массив W(1:JMAX) - веса для аргумента У;

ZPOINT - указатель на массив Z(1:IMAX,1:JMAX) - табличные значения аппроксимируемой функции;

NMAX - некоторая величина, большая, чем максимальная степень X;

MMAX - некоторая величина, большая, чем максимальная степень У;

IMAX - число значений независимых переменных X;

JMAX - число значений независимых переменных У;

Выходные параметры:

BETAPOINT - указатель на массив BETA(1:NMAX,1:MMAX) - мера улучшения, полученная посредством присоединения члена (X**N)*(У**M):

BETA(N,M)=U(N)*W(M)*X(Z(N,M)**2-(Z(N,M)-ZCOMP(N,M))**2)

PHIPOINT - указатель на массив PHI(1:NMAX,1:MMAX) полиномиальных коэффициентов при членах (X**N)*(У**M);

ZCOMPPOINT - указатель на массив ZCOMP(1:IMAX,1:JMAX) из зависимых переменных, вычисляемых для контроля по полученному многочлену во всех узлах рассматриваемой сетки;

$$MINSQD = \left[\frac{\sum_{I,J}^{IMAX,JMAX} U(I)*W(I)*Z(I,J)^{**}2 - \sum_{N,M}^{NMAX,MMAX} BETA(N.M)}{IMAX*JMAX} \right]^{-1/2};$$

$$MINSQDCOMP = \left[\frac{\sum_{I,J}^{IMAX,JMAX} U(I)*W(I)*(Z(I,J)-ZCOMP(I,J)**2)}{IMAX*JMAX} \right]^{-1/2};$$

$$SUMDIFCOMP = \frac{\sum_{I,J}^{IMAX,JMAX} \left| Z(I,J)-ZCOMP(I,J) \right|}{IMAX*JMAX};$$

MAXDIFCOMP = MAX(Z(I,J)-ZCOMP(I,J)) ;

MEANX - среднеарифметическое значение X;
MEANУ - среднеарифметическое значение У;
MEANZ - среднеарифметическое значение Z;

Основными результатами являются массив PHI и величины MEANX, MEANУ, MEANZ.

Расчет аппроксимирующего значения функции Z для текущих X,У следует производить по формуле:

$$Z = MEANZ + \sum_{N}^{NMAX} \sum_{M}^{MMAX} PHI(N,M) * (X-MEANX)**N * (У-MEANУ)**M;$$

Описание в PL/1:

```
DCL SURFACEFIT ENTRY(PTR,PTR,PTR,PTR,PTR,
                    FIXED,FIXED,FIXED,FIXED,
                    PTR,PTR,PTR,
                    FLOAT,FLOAT,FLOAT,FLOAT
                    FLOAT,FLOAT,FLOAT);
```

Пример обращения:

```
CALL SURFACEFIT(ADDR(X),ADDR(U),ADDR(У),ADDR(W),ADDR(Z),
                NMAX,MMAX,IMAX,JMAX,
                ADDR(BETA),ADDR(PHI),ADDR(ZCOMP),
                MINSGD,MINSGDCOMP,SUMDIFCOMP,MAXDIFCOMP,
                MEANX,MEANУ,MEANZ);
```

Источник: М.И.Агеев. Алгоритм 164a.

5.20. Процедура-функция аппроксимации функции от двух переменных

```
INTERMS: PROCEDURE (X,У,X0,У0,DELTAX,DELTAУ,PPOINT,M,N)
                                    RETURNS(FLOAT);
```

X,У - координаты исследуемой точки;
X0,У0 - начальные значения независимых переменных в таблице;
DELTAX, DELTAУ - шаг дискретности таблицы по X и по У соответственно;
PPOINT - указатель на таблицу значений функции P(0:M,0:N);
M,N - размерность P;

Процедура осуществляет аппроксимацию функции f(x,y) по таблице ее значений f(i,j)=f(X0+I*DELTAX,У0+J*DELTAУ), 0<=I<=M, 0<=J<=N, заданных в некотором массиве P.

Описание в PL/1:

```
DCL INTERMS ENTRY (FLOAT,FLOAT,FLOAT,FLOAT,
                   FLOAT, FLOAT,PTR,FIXED,FIXED)  RETURNS(FLOAT);
```

Пример обращения:

```
F=INTERMS(X,У,X0,У0,DELTAX,DELTAУ,ADDR(P),M,N);
```

5.21. Процедура расчета коэффициентов интерполяции полиномом

```
ИНТКОЕФ1: PROC(XPOINT,FPOINT,N,П,XX,З,CPOINT);
```

XPOINT - указатель на таблицу аргументов X(0:N);
FPOINT - указатель на таблицу функции F(0:N);
N - длина таблиц X и F;
П - порядок полинома ;
XX - значение аргумента, для которого требуется построить интерполяционный полином.
З - индекс первого из П+1 интерполяционных узлов, по которым будет строиться интерполяционный полином, определяется в процессе работы процедуры.
CPOINT - указатель на массив коэффициентов интерполяции C(0:П);

Процедура ИНТКОЕФ1 вычисляет коэффициенты полинома

$$P(X)=\sum_{i=0}^{П} Ci * X **(П-i)$$

для заданного аргумента XX, удовлетворяющему условию
X(0) <= XX < X(0);

Описание в PL/1:

```
DCL ИНТКОЕФ1 ENTRY(PTR,PTR,FIXED,FIXED,
                   FLOAT,FIXED,PTR);
```

Пример обращения:

```
CALL ИНТКОЕФ1(ADDR(X),ADDR(F),N,П,XX,З,ADDR(C));
```

6. ОПЕРАЦИИ С МАССИВАМИ

6.1. Процедура-функция выбора максимального элемента из массива

```
MAXIM: PROC(XPOINT,M,N) RETURNS(FLOAT);
```

XPOINT - указатель на входной массив X(K:P);
M и N - начало и конец части массива X, из которой требуется выбрать максимальный элемент (K<=M<=N<=P);

Описание в PL/1:

```
DCL MAXIM ENTRY(PTR,FIXED,FIXED) RETURNS(FLOAT);
```

Пример обращения: `MAKS=MAXIM(ADDR(X),13,113);`

6.2. Процедура-функция выбора максимального по модулю элемента из массива

```
ABSMAX: PROC(XPOINT,M,N) RETURNS(FLOAT);
```

XPOINT - указатель на входной массив X(K:P);
M и N - начало и конец части массива X, из которой требуется выбрать максимальный по модулю элемент (K<=M<=N<=P);

Описание в PL/1:

```
DCL ABSMAX ENTRY(PTR,FIXED,FIXED) RETURNS(FLOAT);
```

Пример обращения: `ABSMAKS=ABSMAX(ADDR(X),13,113);`

6.3. Процедура поиска максимального числа в массиве с запоминанием его места

```
МАКС: PROC(XPOINT,П,У,I);
```

XPOINT - указатель на входной массив X(1:П);
П - длина исследуемой части массива X;
У - результат;
I - номер максимального числа У в массиве X;

Описание в PL/1: `DCL МАКС ENTRY(PTR,FIXED,FLOAT,FIXED);`

Пример обращения: `CALL МАКС(ADDR(X),13,У,I);`

6.4. Процедура-функция выбора минимального элемента из массива

MINIM: PROC(XPOINT,M,N) RETURNS(FLOAT);

XPOINT - указатель на входной массив X(K:P);
M и N - начало и конец части массива X, из которой требуется выбрать минимальный элемент (K<=M<=N<=P);

Описание в PL/1:
DCL MINIM ENTRY(PTR,FIXED,FIXED) RETURNS(FLOAT);

Пример обращения: M=MINIM(ADDR(X),13,113);

6.5. Процедура-функция выбора минимального по модулю элемента из массива

ABSMIN: PROC(XPOINT,M,N) RETURNS(FLOAT);

XPOINT - указатель на входной массив X(K:P);
M и N - начало и конец части массива X, из которой требуется выбрать минимальный по модулю элемент (K<=M<=N<=P);

Описание в PL/1:
DCL ABSMIN ENTRY(PTR,FIXED,FIXED) RETURNS(FLOAT);

Пример обращения: ABSМИН=ABSMIN(ADDR(X),13,113);

6.6. Процедура поиска минимального числа в массиве с запоминанием его места

МИН: PROC(XPOINT,П,У,I);

XPOINT - указатель на входной массив X(1:П);
П - длина исследуемой части массива X;
У - результат;
I - номер максимального числа У в массиве X;

Описание в PL/1: DCL МИН ENTRY(PTR,FIXED,FLOAT,FIXED);

Пример обращения: CALL МИН(ADDR(X),13,У,I);

6.7. Процедура нахождения минимального и максимального значений в массиве

MINMAX: PROC(APOINT,X,У,П);

APOINT - указатель на исходный массив A(1:П);
X - результат (min);
У - результат (max);
П - длина исследуемой части массива X;

Описание в PL/1: DCL MINMAX ENTRY(PTR,FLOAT,FLOAT,FIXED);

Пример обращения: CALL MINMAX(ADDR(X),X,У,123);

6.8. Процедура расположения чисел в массиве в возрастающем порядке

ВОЗР: PROC(APOINT,П);

APOINT - указатель на исходный массив чисел A(1:П);
П - длина массива A;

Результат работы процедуры остается в массиве A;

Описание в PL/1: DCL ВОЗР ENTRY(PTR,FIXED);

Пример обращения: CALL ВОЗР(ADDR(A),113);

6.9. Процедура расположения чисел в массиве в убывающем порядке

УБЫВ: PROC(APOINT,П);

APOINT - указатель на исходный массив чисел A(1:П);
П - длина массива A;

Результат работы процедуры остается в массиве A;

Описание в PL/1: DCL УБЫВ ENTRY(PTR,FIXED);

Пример обращения: CALL УБЫВ(ADDR(A),113);

6.10. Процедура упорядочения массива

QUICKERSORT: PROC(APOINT,J);

APOINT - указатель на исходный массив чисел A(1:J);
J - число элементов в массиве A;

Время работы процедуры зависит от J примерно как C*J*LN(J). Если процедуру применять к массиву A(K:L), то упорядочатся элементы A(I) для 1<=I<=J (K<=1<=J<=L).

Описание в PL/1: DCL QUICKERSORT ENTRY(PTR,FIXED);

Пример обращения: CALL QUICKERSORT(ADDR(A),113);

Источник: Журнал "Communication of the ACM". Алгоритм 271.

7. МЕТОД МОНТЕ-КАРЛО

7.1. Процедура построения вероятностных характеристик вектора, являющегося заданной функцией случайного вектора, имеющего заданные характеристики

```
MONTECARLO: PROCEDURE
                (KSIPOINT,N,MKSIPOINT,SIGMAKSIPOINT,
                   KKSIPOINT,F,ETAPOINT,K,METAPOINT,
                 SIGMAETAPOINT,KETAPOINT,AKPOINT,NN,L);
```

KSIPOINT – указатель на случайный вектор KSI(1:N), значения которого получаются в теле процедуры по заданным вероятностным характеристикам MKSI, SIGMAKSI, KKSI. Перед первым обращением переменной KSI(1), следует присвоить «начальное» псевдослучайное число – целое нечетное значение из интервала 1 <= KSI(1) <=67108863,

K - размерность вектора MKSI,

N - размерность вектора KSI

MKSIPOINT- указатель на вектор MKSI(1:K), являющийся результатом преобразования с помощью процедуры F

SIGMAKSIPOINT – указатель на массив SIGMAKSI(1:N) заданных среднеквадратичных отклонений

KKSIPOINT – указатель на массив KKS(1:N*(N-1)/2) заданных коэффициентов корреляции координат KSI k_{12} k_{13} k_{14} …k_{23} k_{n-1n} после выхода из процедуры MONTECARLO в массивах MKSI SIGMAKSI KKSI получаются соответствующие эмпирические оценки характеристик KSI рассчитанные (для контроля) по всем прошедшим реализациям (этих реализаций будет NN*L штук, смысл величин NN L будет указан ниже)

F, - процедура, преобразующая KSI в ETA ее описание должно иметь вид

```
F:PROC(KSIPOINT,ETAPOINT); ….. END;
```

ETAPOINT – указатель на вектор являющийся результатом преобразования,

NN – заданное количество реализаций KSI («серия» реализаций) ,которое обрабатывается при одном обращении к MONTECARLO

METAPOINT, SIGMAETAPOINT,KETAPOINT - указатели на массивы в которых получаются статистические оценки KSI расположенные в том же порядке

AKPOINT – указатель на рабочий массив AK(1:n,1:N),

L – автоматический счетчик серий. Перед входом в цикл обращений к MONTECARLO (перед первым обращением) нужно присвоить нулю.

При работе цикла в массивах MKSI,SIGMAKSI KKSIPOINT, META,

SIGMAETA, KETA получаются оценки рассчитанные по нарастающему количеству серий, т.е. всего по NN*L реализациям, если было L обращений к MONTECARLO.

Цикл целесообразно организовать так, чтобы вывод результатов происходил после каждого обращения к процедуре (NN всегда можно задать таким, чтобы выводы шли не слишком часто). Тогда цикл можно «замкнуть на бесконечность» и счет задачи можно будет прекратить в любой момент без потери результатов.

Матрица

1 K_{12} ….K_{1n}

K_{12} 1 ….K_{2n}

…..

K_{1n} $k_{2\,n}$ ….1

Должна быть положительно определенной, иначе будет исключение при извлечении корня.

Описание в PL/1:

```
MONTECARLO: ENTRY(PTR,FIXED,PTR,PTR,PTR,
                  ENTRY(PTR,PTR),
            PTR,FIXED,PTR,PTR,PTR,PTR,FIXED,FIXED);
```

Пример обращения:

```
CALL IMONTECARLO: (KSIPOINT,N,MKSIPOINT,SIGMAKSIPOINT,
                   KKSIPOINT,F,ETAPOINT,
                   K,METAPOINT,SIGMAETAPOINT,
                   KETAPOINT,AKPOINT,NN,L);
```

Источник: Более подробные сведения о процедурах метода Монте-Карло можно получить в следующих работах

Любимов Ю.К. «Процедуры получения псевдослучайных чисел». ОТИИ 1969

Кудрявцев А.Д. Лейтес В.Л. Любимов Ю.К. «Применение метода Монте-Карло и интегрирование дифференциальных уравнений с переменным шагом» ОТИИ 1969

Любимов Ю.К. Газиева А.А. Горельков А.Л. «Применение метода Монте-Карло и интегрирование дифференциальных уравнений с переменным шагом» ОТИИ 1969

7.2. Процедура-функция получения случайных чисел, равномерно распределенных на отрезке [-1,1]

UR: PROCEDURE(X) RETURNS(FLOAT);

X – перед входом в цикл получения случайных чисел должно быть присвоено «начальное число» целое нечетное число в пределах 1-67108863. при различных начальных числах получаются различные последовательности случайных чисел в цикле величина меняется автоматически и основная программа как правило не должна ее менять

Изменение допускается только в том случае, когда требуется обновить величину для исключения повторения случайных чисел.

Для получения равномерно распределенного числа на произвольном отрезке A B обращение нужно написать в виде

Y=1/2(B-A)*(UR(X)+1)+A

Для отрезка 0 1 преобразование имеет вид Y=1/2*(ur(x)+1)

Описание в PL/1: DCL UR ENTRY(FIXED) RETURNS(FLOAT);

Пример обращения: Y=UR(X);

7.3. Процедура получения нормально и равномерно распределенных случайных чисел

UNR: PROCEDURE(X,A,V);

При каждом обращении переменная V получает значение очередного нормально распределенного числа с математическим ожиданием M(V)=0 и дисперсией Д(V)=1 переменная A получает значение равномерно распределенного числа на отрезке -1 1

X - перед входом в цикл получения случайных чисел должно быть присвоено «начальное число» целое нечетное число в пределах 1-67108863.

При различных начальных числах получаются различные последовательности случайных чисел в цикле величина меняется автоматически.

Описание в PL/1: DCL UNR ENTRY (FIXED,FLOAT,FLOAT);

Пример обращения: UNR(X,Y,Z);

7.4. Процедура-функция генератор псевдослучайных чисел с нормальным распределением

RND: PROCEDURE (RANDOM) RETURNS(FLOAT);

Процедура-функция RND с помощью процедуры-функции RANDOM Вырабатывает очередное значение нормально распределенной величины с M=0 и σ=1

Для использования необходимо описать и вызвать процедуру RANDOM и присвоить начальное значение величине Y (см. описание RANDOM) обращение к процедуре-функции имеет вид:

z=M0+rnd(random(y))* σ при M0 не равном нулю и σ не равном 1

z=rnd(random(y)) при M0 равном нулю и σ равном 1

Описание в PL/1:

RND ENTRY(ENTRY(FLOAT) RETURNS(FLOAT))RETURNS(FLOAT);

Источник: процедура-функция взята из работы Д. И. Голенко «Моделирование и статистический анализ псевдослучайных чисел на ЭВМ» «Наука» 1965

7.5. Процедура-функция генератор псевдослучайных чисел с равномерным распределением на отрезке (0,1)

RANDOM: PROCEDURE(Y) RETURNS(FLOAT);

Процедура-функция при каждом обращении вырабатывает одно псевдослучайное число на интервале 0 1

Y - перед входом в цикл получения случайных чисел должно быть присвоено «начальное число» целое нечетное число в пределах 1-67108863.

Во избежании циклов в длинных последовательностях чисел значение переменной рекомендуется обновлять.

Процедура вырабатывает числа с M0 равном нулю и σ равном 1.

Описание в PL/1: DCL RANDOM ENTRY(FLOAT) RETURNS(FLOAT);

Пример обращения: X=RANDOM(Y);

7.6. Процедуры построения законов распределения случайной величины

```
PV1: PROCEDURE(KSIPOINT,K,N,APOINT,NN,AAPOINT,
                    BPOINT,MPOINT,SIGMAPOINT);

PV2: PROCEDURE(K,N,APOINT,NN,AAPOINT,BPOINT,
                    MPOINT,SIGMAPOINT);

PV3: PROCEDURE(K,N,APOINT,PPOINT,NN,MPOINT,SIGMAPOINT);
```

Для построения законов распределения векторной случайной величины ξ должно происходить обращение к процедуре PV1 в основном цикле программы после получения каждой очередной реализации ξ после завершения всей работы основного цикла необходимо обратиться либо к процедуре PV2 либо к процедуре PV3.

Процедура PV2 используя информацию накопленную процедурой PV1 вычисляет плотность распределения каждой координаты вектора ξ

Процедура PV3 вычисляет гистограмму (таблицу частот) каждой координаты и ее функцию распределения (интегральный закон) кроме того, PV2, PV3 вычисляют математическое ожидание и дисперсию.

```
PV1: PROCEDURE(KSIPOINT,K,N,APOINT,NN,AAPOINT,
                    BPOINT,MPOINT,SIGMAPOINT);
```

KSIPOINT – указатель на массив KSI(1:K) реализация случайного вектора ξ,

K - размерность KSI

N - автоматический счетчик реализаций перед входом в цикл необходимо присвоить нулю;

APOINT - указатель на массив A(1:K,-1:N+1) для накопления информации о векторе используется далее процедурой Pv2 или PV3.

NN - число частей на которые разбивается каждый отрезок a(i) b(i) при построении плотности (n+1 количество значений в таблице плотности).

AAPOINT, BPOINT - указатели на массивы A(1:K) B(1:K), где A(i) есть начало a B(i) – конец отрезка на котором должны лежать все (или почти все) значения i-той координаты

MPOINT - указатель на массив M(1:K) для накопления сумм реализаций координат.

SIGMAPOINT - указатель на массив SIGMA(1:K) для накопления сумм квадратов реализаций ξ;

Обращение к процедуре PV2 имеет смысл только при наличии в основном цикле обращения к PV1.

```
PV2: PROCEDURE(K,N,APOINT,NN,AAPOINT,BPOINT,
                    MPOINT,SIGMAPOINT);
```

В результате обращения к PV2 в элементах A(1:K,0:N) массива A получаются таблицы плотностей вероятности координат ξ.

При фиксированном i строка $A_{i\ 0\text{-}n}$ дает плотность Pi(x) распределения случайной величины KSI(i) в точках A_i, A_{i+dx},…$A_{i+ndx}=B_i$

A(i,-1)- количество значений KSI(i) попавших левее точки A(i)

A(i,n+1) – количество значений KSI(i) попавших правее точки B(i)

MPOINT - указатель на вектор M(1:K) математических ожиданий координат ξ;

SIGMAPOINT – указатель на вектор SIGMA(1:K) указатель на вектор среднеквадратичных отклонений координат ξ;

Обращение к PV3 имеет смысл только при наличии в основном цикле обращения к PV1.

```
PV3: PROCEDURE(K,N,APOINT,PPOINT,NN,MPOINT,SIGMAPOINT);
```

В результате обращения к PV3 в элементах A(i:K,0:N) массива A получаются гистограммы координат ξ при этом:

В A(i,-1) получается вероятность того, что

KSI(i)<a(i)-d(i)/2 где d(i)=(b(i)-a(i))/n

В A(I,j) j-0,1,…n вероятность того, что

ABS(KSI(i)-(a(i)+j*d(i))) <d(i)/2

PPOINT – указатель на массив P(1:k,0:n) где получаются интегральные законы распределения. Именно P(i,j) есть сглаженная (т.е. усредненная по двум прилегающим отрезкам) вероятность того, что

KSI(i)<a(i)+j*d(i) j=0,1,…n i=1,2,….k

MPOINT – указатель на массив M(1:K) математического ожидания координат ξ.

SIGMAPOINT - указатель на массив SIGMA(1:K) для среднеквадратичных отклонений

Примечание: фактические параметры в обращении к PV2 и PV3 должны совпадать с соответствующими параметрами обращения к PV1.

Описание в PL/1:

```
DCL
PV1 ENTRY(PTR,FIXED,FIXED,PTR, FIXED,PTR,PTR,PTR,PTR),
PV2 ENTRY(FIXED,FIXED,PTR,FIXED,PTR,PTR,PTR,PTR),
PV3 ENTRY(FIXED,FIXED,PTR,PTR,FIXED,PTR,PTR);
```

7.7. Процедура построения гистограммы случайной величины с неизвестными пределами

```
PVAR: PROCEDURE(KSI,NN,N,GPOINT,M,SIGMA,MIN,MAX);
```

KSI – случайная величина (очередная реализация)

NN – автоматический счетчик реализаций перед входом в цикл необходимо присвоить нулю;

N - число делений неизвестного отрезка на котором будут расположены все реализации;

GPOINT – указатель на массив G(0:n-1) для гистограммы (в ячейке G(i) получается вероятность попадания KSI в интервал MIN+ih MIN+(i+1)h, где MIN текущее начало отрезка определения KSI автоматически определяемой процедурой, а h=(MAX-MIN)/n

M - сумма реализаций;

SIGMA – сумма квадратов реализаций по этим величинам можно рассчитать оценки математического ожидания и среднеквадратичного отклонения M=M/n, σ=sqrt(σ-m**2)

MIN, MAX - текущие концы интервала на котором построена гистограмма (выходные величины);

Обращение к процедуре PVAR должно находиться в цикле перебора реализаций исследуемой скалярной случайной величины.

Описание в PL/1:

```
DCL PVAR ENTRY(FLOAT,FIXED,FIXED,PTR,FLOAT,
                    FLOAT,FLOAT,FLOAT);
```

Пример обращения:

```
CALL PVAR(KSI,NN,N,GPOINT,M,SIGMA,MIN,MAX);
```

7.8. Процедуры получения корреляционной матрицы случайного вектора

```
MK1: PROCEDURE (KSIPOINT,MPOINT,SIGMAPOINT,
                KPOINT,N,N2);
MK2: PROCEDURE (MPOINT,SIGMAPOINT,KPOINT,N,N2);
```

Для получения корреляционной матрицы случайного вектора ξ по его реализациям в основном цикле программы должно стоять обращение к процедуре MK1. После завершения работы цикла перебора реализаций должно происходить обращение к процедуре MK2, которая, используя информацию, накопленную MK1, вычисляет коэффициенты корреляции среднеквадратичные отклонения и математические ожидания координат вектора ξ;

```
MK1:PROCEDURE (KSIPOINT,MPOINT,SIGMAPOINT,KPOINT,N,N2)
```

KSIPOINT - указатель на исследуемый случайный вектор (очередная реализация) KSI(1:N)

MPOINT - указатель на массив M(1:N) для накоплений сумм реализаций координат KSI

SIGMAPOINT- указатель на массив SIGMA(1:N) для накопления сумм квадратов реализаций;

KPOINT- указатель на массив K(1:N*(N-1)/2) для накоплений сумм произведений координат ξ;

N - размерность KSI

N2 - автоматический счетчик реализаций перед входом в основной цикл его нужно присвоить нулю;

```
MK2: PROCEDURE (MPOINT,SIGMAPOINT,KPOINT,N,N2);
```

Обращение к процедуре MK2 имеет смысл только при наличии обращения к MK1 в основном цикле. Все параметры имеют тот же смысл, что и в MK1. При работе MK2 значения N, N2 не меняются. В массивах M, SIGMA, K получаются соответственно:

- вектор математических ожиданий координат ξ;
- вектор среднеквадратичных отклонений ξ;
- вектор коэффициентов корреляции координат ξ, расположенных в следующем порядке K_{12} K_{13} K_{1n} K_{23}…. K_{2n}…. K_{n-1n}

Описание в PL/1:

```
DCL
MK1 ENTRY (PTR,PTR,PTR,PTR,FIXED,FIXED),
MK2 ENTRY (PTR,PTR,PTR,FIXED,FIXED);
```

Пример обращения:

```
CALL MK1(KSIPOINT,MPOINT,SIGMAPOINT,KPOINT,N,N2);
CALL MK2(MPOINT,SIGMAPOINT,KPOINT,N,N2);
```

7.9. Процедура табулирования функции нормального распределения

```
NORMAL: PROCEDURE(XPOINT,M,VARS,C,PPOINT);
```

XPOINT - указатель на массив точек деления Xi числовой оси X(1:C)
VARS – дисперсия;
C - число интервалов разбиения без единицы;
PPOINT - указатель на массив вычисляемых значений вероятности P(1:C+1)
Первый и последний интервалы имеют вид -∞ X1 Xc+1 +∞

Процедура вычисляет вероятности Pi попадания нормально распределенной случайной величины в интервалы Xi-1 Xi по формуле

$$Pi = \int_{0}^{Xi} f(x)\,dx - \int_{0}^{Xi-1} f(x)\,dx$$

Где $f(x)=1/(\sigma \sqrt{2\pi})\, e^{-(x-m)^{**}2/(2^{*}\sigma\sigma)}$ - плотность вероятности

Для аппроксимации нормальной функции распределения используется формула Гастингса

$$\Phi(xni)=0{,}5-0{,}5(1+a1xni+a2\,xni^{2}+a3xni^{3}+a4\,xn^{4}+a5\,xni^{5})^{-8}$$

Где
a1=0,9979268 a2=0,04432014 a3=0,00969920
a4=-0,00009862 a5=0,00058155

xni – нормализованное значение xi
xni=(xi-m)/sqrt(vars)

значение P_1 – вероятность попадания в интервал -∞ X_1
значение P_{c+1} – вероятность попадания в интервал X_{c+1} +∞

Описание в PL/1:

```
DCL NORMAL ENTRY(PTR,FLOAT,FLOAT,FIXED,PTR);
```

Пример обращения:

```
CALL NORMAL(XPOINT,M,VARS,C,PPOINT);
```

Источник: М.И.Агеев. Алгоритм 185.

8. ЭКСТРЕМАЛЬНЫЕ ЗАДАЧИ

8.1. Процедура отыскания экстремума функции одной переменной

EXTR: PROC(XY,IN,FUN,PI1,PI2);

XY - массив размерности 1:2;
IN - массив размерности 1:4;
PI1,PI2 - простые переменные, атрибут FIXED(7);
XY(1)=X0 начальное приближение по аргументу;
XY(2)=F(X0) значение функции в точке X0 (необязательный параметр, регулируется параметром PI2);
IN(1) - начальный шаг по аргументу;
IN(2) - заданная точность;
IN(3),IN(4) - начало и конец интервала, на котором ищется экстремум;
PI1= 1 при поиске максимума, 0 при поиске минимума;
PI2= 1 если заданы X0 и F(X0), 0 если задано только X0;
FUN - имя процедуры-функции, вычисляющей значение F(X)

После выхода из процедуры XY(1) и XY(2) являются координатами точки экстремума (X1,F(X1)).

Описание в PL/1:

```
DCL EXTR ENTRY ((1:2)FLOAT,(1:4)FLOAT,
                ENTRY(FLOAT) RETURNS (FLOAT),
                FIXED(7), FIXED(7));
```

Пример обращения: CALL EXTR (XY,IN,F1,PI1,PI2);

8.2. Процедура минимизации функции нескольких переменных методом скорейшего спуска

```
STEEP1: PROCEDURE(LBPOINT,UBPOINT,DXPOINT,N,EPS,
                  RELAX,DXMAX,ETA,PSI, PMAX,ZETA,FUNK,
                  XMINPOINT,FMIN);
```

Входные параметры:

LBPOINT - указатель на вектор LB[1:N] нижних границ независимых переменных.

UBPOINT - указатель на вектор UB[1:N] верхних границ независимых переменных.

DXPOINT - указатель на массив DX[1:N] приращений независимых переменных; приращения выбираются таким образом, чтобы выполнялось неравенство:

$$ETA < \left| \frac{f(x+dx)-f(x-dx)}{f(x)} \right| < 100\ ETA$$

N - число независимых переменных;

EPS - точность вычисления FMIN;

RELAX - параметр релаксации; подбирается опытным путем из [0,1]

DXMAX - максимально допустимое значение приращений по всем DX(I);

ETA - мера относительной ошибки вычисления функции;

PSI - минимальное допустимое значение $|f(x)|$ в неравенстве;

PMAX - максимально допустимое значение коэффициента увеличения числа (1<=p<=10)

ZETA - минимально допустимое значение $|Xi|$ при выборе продвижения по переменной Xi;

FUNK - имя процедуры, вычисляющей значения минимизируемой функции от переменных массива X;

Выходные параметры:

XMINPOINT - указатель на XMIN(1:N) массив координат вычисляемой точки минимума; при обращении к процедуре массив должен содержать координаты точки, от которой начинается поиск минимума;

FMIN - значение функции в точке XMIN;

Описание в PL/1:

```
DCL STEEP1 ENTRY (PTR,PTR,PTR,FIXED,FLOAT,FLOAT,FLOAT,
                  FLOAT,FLOAT,FLOAT,FLOAT,
                  ENTRY(PTR) RETURNS (FLOAT),
                  PTR,FLOAT);
```

Пример обращения:

```
CALL STEEP1(ADDR(LB),ADDR(UB),ADDR(DX),4,0.001,
            RELAX,DXMAX,ETA,PSI,PMAX,ZETA,FUNK,
            ADDR(XMIN),FMIN);
```

Источник: М.И.Агеев. Алгоритм 203а.

8.3. Процедура компромиссного варианта для минимизации функции нескольких переменных методом скорейшего спуска

```
STEEP3: PROCEDURE(LBPOINT,UBPOINT,DXPOINT,N,EPS,
RELAX,DXMAX,ETA,PSI,PMAX,ZETA,FUNK,XMINPOINT,FMIN);
```

Входные параметры:

LBPOINT - указатель на вектор LB[1:N] нижних границ независимых переменных

UBPOINT - указатель на вектор UB[1:N] верхних границ независимых переменных

DXPOINT - указатель на массив DX[1:N] приращений независимых переменных; приращения выбираются таким образом, чтобы выполнялось неравенство:

$$ETA < \left| \frac{f(x+dx)-f(x-dx)}{f(x)} \right| < 100\ ETA$$

N - число независимых переменных;

EPS - точность вычисления FMIN;

RELAX - параметр релаксации; подбирается опытным путем из [0,1]

DXMAX - максимально допустимое значение приращений по всем DX(I);

ETA - мера относительной ошибки вычисления функции;

PSI - минимальное допустимое значение $|f(x)|$ в неравенстве;

PMAX - максимально допустимое значение коэффициента увеличения числа (1<=p<=10)

ZETA - минимально допустимое значение $|Xi|$ при выборе продвижения по переменной Xi;

FUNK - имя процедуры, вычисляющей значения минимизируемой функции от переменных массива X;

Выходные параметры:

XMINPOINT - указатель на XMIN(1:N) массив координат вычисляемой точки минимума; при обращении к процедуре массив должен содержать координаты точки, от которой начинается поиск минимума;
FMIN - значение функции в точке XMIN;

Описание в PL/1:

```
DCL STEEP3 ENTRY (PTR,PTR,PTR,FIXED,FLOAT,FLOAT,FLOAT,
                  FLOAT,FLOAT,FLOAT,FLOAT,
                  ENTRY(PTR) RETURNS (FLOAT),
                  PTR,FLOAT);
```

Пример обращения:

```
CALL STEEP3(ADDR(LB),ADDR(UB),ADDR(DX),4,0.001,
            RELAX,DXMAX,ETA,PSI,
            PMAX,ZETA,FUNK,ADDR(XMIN),FMIN);
```

Источник: М.И.Агеев. Алгоритм 205a.

8.4. Процедура минимизации функции нескольких переменных методом прямого спуска

DIRECT (K,RH0,DELTA,MAX,PSIPOINT,D,SPSI,CORR);

K - количество переменных;
DELTA - минимально допустимый шаг поиска; окончание счета, если шаг меньше DELTA;
MAX - максимально допустимое количество вычислений функции.
PSIPOINT - указатель на PSI(1:K) начальная точка поиска, после работы процедуры в массиве PSI(1:K) найденная точка экстремума.
D - начальный шаг поиска;
SPSI - значение функции в точке PSI(1:K);
CORR=1 если выход из процедуры по минимуму, и 0 если выход по ограничению количества проходов вычисления функции;

В вызывающей программе должно быть описание

```
DCL PRCDIRECT ENTRY (PTR) EXT VARIABLE RETURNS (FLOAT);
```

Перед обращением к DIRECT параметру PRCDIRECT необходимо присвоить имя процедуры, вычисляющей функцию.

Описание в PL/1:

```
DCL DIRECT ENTRY(FIXED,FLOAT,FLOAT,FIXED,PTR,FLOAT,
                 FLOAT,FIXED(7));
```

Пример обращения:

```
CALL DIRECT(K,RH0,DELTA,MAXADDR(PSI),D,SPSI,CORR);
```

9. ОПЕРАЦИИ НАД РЯДАМИ И КОНЕЧНЫМИ СУММАМИ

9.1. Процедура-функция суммирования значений арифметического выражения

SUM(APOINT,M,N);

Процедура вычисляет сумму значений из массива A(I) при I меняющимся от M до N с шагом 1.

APOINT указатель на массив A(X,Y);

M, N – начальное и конечное значение диапазона.

Описание в PL/1:

```
DCL SUM ENTRY (PTR,FIXED,FIXED) RETURNS (FLOAT);
```

Пример обращения:

```
CALL SUM(ADDR(LB),50,60);
```

9.2. Процедура суммирования рядов Фурье

FOURIER: PROCEDURE(X,R,W,N,A,B);

X – может быть выражением, зависящим от R либо X(R) как элемент массива, если коэффициенты Фурье задаются массивом X(0:N);

R – индекс коэффициента Фурье

W - аргумент

N,- общее число членов ряда

A,B - результат

$$A=\sum_{r=0}^{n-1} Xr \cos(RW); \quad B=\sum_{r=0}^{n-1} Xr \sin(RW);$$

Описание в PL/1:

```
FOURIER: ENTRY (FLOAT,FIXED,FLOAT,FIXED,FLOAT,FLOAT);
```

Пример обращения: FOURIER(X,R,W,N,A,B);

Источник: М.И.Агеев. Алгоритм 128а.

9.3. Процедура деления степенного ряда на степенной ряд

SERDIV: PROCEDURE(N,GPOINT,HPOINT);

N - число коэффициентов степенных рядов g h
GPOINT - указатель на массив делителя;
HPOINT - указатель на массив коэффициентов делимого;
Результат получается в массиве h.
Процедура вычисляет первые n коэффициентов степенного ряда H(z)/G(z), если заданы первые n коэффициентов степенных рядов

G(z)=g1+g2*z+g3*z**2 ….+g n** Zn-1
H(z)=h1+h2*z+h3*z**2 +h*n**Zn-1

Предполагается, что g1 не равно нулю.

Описание в PL/1: DCL SERDIV ENTRY(FIXED,PTR,PTR);

Пример обращения: CALL SERDIV(N,GPOINT,HPOINT);

Источник: М.И.Агеев. Алгоритм 131а.

9.4. Процедура преобразования полинома n-й степени

POLYX: PROCEDURE (A,B,CPOINT,N,DPOINT);

A,B - коэффициенты преобразования X=At+B,
CPOINT- указатель на массив C(0:N) коэффициентов исходного полинома, $p(x)=c_0+c_1x+\ldots+c_nx_n$;
N - степень полинома p(x),
DPOINT - указатель на массив D(0:n) коэффициентов преобразованного полинома $P(t)=d_0+d_1t+\ldots d_nt_n$

Описание в PL/1:
DCL POLYX ENTRY (FLOAT,FLOAT,PTR,FIXED,PTR);

Пример обращения: CALL POLYX (A,B,CPOINT,N,DPOINT);

Источник: М.И.Агеев. Алгоритм 29а.

9.5. Процедура суммирования по Эйлеру

EULER: PROCEDURE(FCT,EPS,TIM,SUM);

FCT – процедура-функция,
EPS - точность суммирования,
TIM- число контрольных членов для проверки на точность,
S – результат;

Процедура вычисляет сумму

$$Sum = \sum_{I=0}^{\infty} fct(i)$$

Суммирование прекращается как только TIM подряд стоящих членов преобразования по Эйлеру суммы будут по модулю меньше EPS.

Процедура особенно эффективна в случае медленно сходящихся и расходящихся рядов.

Описание в PL/1:

```
DCL EULER ENTRY(ENTRY(FIXED) RETURNS(FLOAT),
                FLOAT,FIXED,FLOAT);
```

Пример обращения: CALL EULER(FCT,EPS,TIM,SUM);

Источник: М.И.Агеев. Алгоритм 8а.

9.6. Процедура вычисления преобразования Фурье от комплексной дискретной функции

```
REALCOMPLEXTRANSFORM:
                PROCEDURE(APOINT,BPOINT,M,INVERSE);
```

APOINT - указатель на массив вещественных частей компонент временного ряда A(0:n-1),

BPOINT - указатель на массив мнимых частей компонент временного ряда B(0:n-1),

M - степень ряда;

INVERSE - логическая переменная определяющая тип преобразований.

Прямое преобразование т.е. вычисление n значений комплексного спектра осуществляется при значении ‘0’b.

Результат преобразований ставится в те же массивы A, B

Если требуется по комплексному спектру определить временной ряд то значение ‘1’b.

Процедура реализует быстрое преобразование Фурье временного ряда {aj+ibj}, состоящего из N=2**M комплексных чисел (M - целое число).

$$P_j+iQ_j=1/\sqrt{n}\sum_{k=1}^{n-1}(a_k+ib_k)\,L^{i2\pi jk/n} \qquad j=0,\ldots..n-1$$

Описание в PL/1:

```
DCL REALCOMPLEXTRANSFORM ENTRY(PTR,PTR,FIXED,BIT(1));
```

Пример обращения:

```
CALL REALCOMPLEXTRANSFORM(APOINT,BPOINT,M,INVERSE);
```

10. ПРИКЛАДНЫЕ ПРОЦЕДУРЫ

10.1. Процедура вычисления параметров стандартной атмосферы

ATM(H,УPOINT);

H – задаваемая высота в метрах.
УPOINT – указатель на массив У(1:3) возвращаемых ответов: давление, температура, плотность.

Описание в PL/1: DCL ATM ENTRY(FLOAT,PTR);

Пример обращения: CALL ATM(3500,ADDR(X));