МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Томский государственный университет систем управления и радиоэлектроники (ТУСУР)
Кафедра моделирования и системного анализа (МиСА)

В.Г. Баранник, Е.В. Истигечева

ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА

Учебно-методическое пособие

Томск 2014

Баранник В.Г., Истигечева Е.В. Вычислительная математика / Учебно-методическое пособие – Томск: Томский государственный университет систем управления и радиоэлектроники, Кафедра моделирования и системного анализа, 2014. – 27 с.

Содержание

Введение

Курс «Вычислительная математика» рассматривает алгоритмы элементарной алгебры, которые обычно не включаются в библиотеки математического обеспечения ЭВМ, хотя довольно часто оказываются необходимыми при решении соответствующих задач. Задача курса состоит в изучении вычислительных алгоритмов задач высшей математики и освоении элементарных навыков моделирования.

Задачей настоящих методических указаний является помощь студентам очной формы обучения при изучении курса «Вычислительная математика».

Перечень лабораторных работ

Лабораторная работа № 1.(8 часов). Решение уравнений с одной переменной.

Лабораторная работа № 2.(6 часа). Решение СЛАУ.

Лабораторная работа № 3.(4 часа). Интерполирование функций сплайнами.

Лабораторная работа № 4.(6 часа). Приближенное вычисление определенных интегралов.

Лабораторная работа № 5.(4 часа). Приближенное вычисление кратных интегралов.

Лабораторная работа № 6.(8 часа). Метод Рунге-Кутта и Эйлера.

Лабораторная работа № 1. Решение уравнений с одной переменной.

Цель работы – закрепление практических навыков решения задач на алгоритмическом языке Pascal, необходимых для выполнения лабораторных работ по дисциплине.

Методы решения нелинейных уравнений

1. Метод хорд

а) Если $f(b)*f''(x)>0$ на $[a,b]$, то

$$x_{n+1} = x_n - \frac{f(x_n)}{f(b) - f(x_n)}(b - x_n)$$, при этом $x_0=a$.

б) Если $f(a)*f''(x)>0$ на $[a,b]$, то

$$x_{n+1} = a - \frac{f(a)}{f(x_n) - f(a)}(x_n - a)$$, при этом $x_0=b$.

Формальные параметры процедуры. Входные: a, b (тип ***real***) - отрезок, на котором ищется корень; *eps* (тип ***real***) - точность вычисления корня; *k* (тип ***integer***) - разрешенное число итераций; *FUNC* - внешняя процедура-функция. *Выходные: k* (тип ***integer***) - количество выполненных итераций; *x* (тип ***real***) - найденное значение корня с заданной точностью *eps*.

```
PROCEDURE HORD (A,B,EPS:REAL; IT:INTEGER;
                        VAR X : REAL; VAR K:INTEGER);
VAR X1,X2,X3 : REAL; K1 : INTEGER;
BEGIN
    K := 1;
    X1 := FUNC(A);
    X2 := FUNC(B);
    X3 := B - X2*(B-A)/(X2-X1);
    IF SIGN(X2)=SIGN(FUNC(X3)) THEN
        K1:=1
    ELSE
    BEGIN
        K1:=2;
        X1:=X2;
    END;
    REPEAT
        INC (K);
        X := X3;
        CASE K1 OF
            1: X2:= X3-FUNC(X3)*(X3-A)/(FUNC(X3)-X1);
            2: X2:= X3-FUNC(X3)*(B-X3)/(X1-FUNC(X3));
        END;
        X3 := X2;
```

```
    UNTIL (K>IT) OR (ABS(X-X2)<EPS);
END.
```

2. Метод Ньютона (метод касательных)

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_0)}.$$

Формальные параметры процедуры. *Входные*: *x*0 (тип ***real***) - начальное приближение; *n* (тип ***integer***) - максимально допустимое количество итераций; *eps* (тип ***real***) - значение в условии окончания итерационного процесса; *func* - имя внешней подпрограммы-функции (тип ***real***), по которой вычисляется значение функции *F*(*x*); *funcp* - имя внешней подпрограммы-функции (тип ***real***), по которой вычисляется значение производной функции *F'*(*x*). *Выходные: i* (тип ***integer***) - количество выполненных итераций; *ih* (тип ***integer***) - указатель причины окончания процесса вычислений: если *ih*=0, то корень уравнения найден с заданной точностью; если *ih*=1, то итерации не сходятся к корню уравнения, и это значит, что на отрезке [*a*,*b*] не выполняется условие теоремы; если *ih*=2, то количество совершенных итераций превзошло максимально допустимое число *N*, что говорит либо о плохом начальном приближении, либо о малости величины |*F'*(*x*)| в окрестности корня; *x* (тип ***real***) - вычисленное значение корня.

```
PROCEDURE NEWT (N: INTEGER; X0,EPS:REAL;
                        VAR I,IH:INTEGER; VAR X:REAL);
VAR DEL,XS,DEL0,Y: REAL;
BEGIN                   DEL0 := 1.E12;
    XS := X0;                   I := 0;
    IH := 0;
    REPEAT                      Y := FUNCP (XS);
        X := XS - FUNC (XS) / Y;
        DEL := ABS ( X - XS);
        IF DEL <= DEL0 THEN
        BEGIN
            IF I>N THEN
            BEGIN
                IH := 2;            EXIT;
            END;
            INC(I);             XS := X;
            DEL0 := DEL;
        END
        ELSE
        BEGIN
            IH := 1;
            EXIT;
        END;
    UNTIL DEL < EPS;
END.
```

3. Метод итераций

Уравнение *f(x)=0* следует привести к виду *x=φ(x)*, например, по формуле:

$$\varphi(x) = x - \frac{f'(x)}{k},$$

причем *k* следует выбрать так, чтобы $|k|>Q/2$, где $Q = \max_{[a,b]} |f'(x)|$ и *k* совпадал бы со знаком $f'(x)$ на интервале [*a;b*], Итерационный процесс сходится при условии $|\varphi'(x)| < 1$ на интервале [*a;b*].

Формальные параметры обеих процедур. Входные: *x*0 (тип ***real***) - начальное приближение корня; *eps* (тип ***real***) - параметр, используемый для окончания итерационного процесса; *ki* (тип ***integer***) - максимальное количество разрешенных итераций; *func* - имя внешней процедуры-функции (тип ***real***), возвращающей значение *w*(*x*). *Выходные:* *x*0 (тип ***real***) - приближенное значение корня, вычисленное с заданной точностью; *k* (тип ***integer***) - количество выполненных итераций; *l* (тип ***integer***) - параметр, контролирующий работу процедуры (только в *iter*2): *l*=0 - вычисления выполнены с заданной точностью и в *x*0 находится значение корня; *l*=1 - вычисления прерваны из-за того, что итерационный процесс стал расходиться (не выполнены условия применимости метода, сформулированные в теореме); *l*=2 - слишком много итераций (больше, чем *ki*). Следует выбрать другое приближение, более близкое к *L* или подобрать иную функцию *w*(*x*).

```
FUNCTION ITER1 (X0: REAL; EPS : REAL;
                                VAR K : INTEGER; KI : INTEGER) : REAL;
VAR X, Y : REAL;
BEGIN  K := 0;
       Y := X0;
       REPEAT          X := Y;
                Y := FUNCI (X);
                INC (K);
       UNTIL (ABS(X-Y) < EPS) OR ( K > KI); ITER1 := X;
END.
```

Но если заранее неизвестно: выполняются условия или нет, то в процедуру надо включить дополнительную проверку, что не намного усложнит программу:

```
FUNCTION ITER2 (X0: REAL; EPS : REAL;
                                VAR K,L : INTEGER; KI : INTEGER) : REAL;
VAR X, Y, EPS1, EPS2 : REAL;
BEGIN   K := 0;
        Y := X0;          L := 0;
        X := Y;
        Y := FUNCI (X);
        K := 1;
        EPS1 := ABS (X-Y);
        REPEAT          X := Y;
                Y := FUNCI (X);
                INC (K);
                EPS2 := ABS (X-Y);
                IF EPS2 > EPS1 THEN L := 1;
                IF K > KI THEN L := 2;
                EPS1 := EPS2;
        UNTIL (EPS2 < EPS) OR ( L<>0 );
        ITER2 := X;
END.
```

Лабораторная работа № 2. Решение СЛАУ

Цель работы – исследование методов решения систем линейных алгебраических уравнений.

1. Формулы Крамера

Пусть дана система *n* уравнений с неизвестными

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1, \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n = b_n. \end{cases}$$

Если определитель отличен от нуля, то её решения находятся по формулам

$$x_1 = \frac{\Delta_{x_1}}{\Delta}, \quad x_2 = \frac{\Delta_{x_2}}{\Delta}, \quad x_3 = \frac{\Delta_{x_3}}{\Delta}, \quad \ldots, \quad x_n = \frac{\Delta_{x_n}}{\Delta},$$

где $\Delta = \begin{vmatrix} a_{11}\, a_{12}\, a_{13} \cdots a_{1n} \\ a_{21}\, a_{22}\, a_{23} \cdots a_{2n} \\ \ldots\ldots\ldots\ldots\ldots \\ a_{n1}\, a_{n2}\, a_{n3} \cdots a_{nn} \end{vmatrix} \neq 0$ – определитель системы,

$\Delta_1, \Delta_2, \ldots, \Delta_n$ – дополнительные определители

$$\Delta_1 = \begin{vmatrix} b_1\, a_{12}\, a_{13} \cdots a_{1n} \\ b_2\, a_{22}\, a_{23} \cdots a_{2n} \\ \ldots\ldots\ldots\ldots\ldots \\ b_n\, a_{n2}\, a_{n3} \cdots a_{nn} \end{vmatrix}, \quad \Delta_2 = \begin{vmatrix} a_{11}\, b_1\, a_{13} \cdots a_{1n} \\ a_{21}\, b_2\, a_{23} \cdots a_{2n} \\ \ldots\ldots\ldots\ldots\ldots \\ a_{nn}\, b_n\, a_{n3} \cdots a_{nn} \end{vmatrix}, \ \ldots, \ \Delta_n = \begin{vmatrix} a_{11}\, a_{12}\, a_{13} \cdots b_1 \\ a_{21}\, a_{22}\, a_{23} \cdots b_2 \\ \ldots\ldots\ldots\ldots\ldots \\ a_{nn}\, a_{n2}\, a_{n3} \cdots b_n \end{vmatrix}$$

2. Схема единственного деления (схема Гаусса) для решения системы уравнений (на примере системы четырёх уравнений)

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + a_{14}x_4 = a_{15}, \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + a_{24}x_4 = a_{25}, \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + a_{34}x_4 = a_{35}, \\ a_{41}x_1 + a_{42}x_2 + a_{43}x_3 + a_{44}x_4 = a_{45}, \end{cases}$$

Формальные параметры процедуры. Входные: n (тип ***integer***) - порядок системы; *a* (тип ***real***) - массив коэффициентов системы размером *n*×*n*; *b* (тип ***real***) - массив-строка свободных членов. *Выходные: x* (тип ***real***) - массив-строка, в который помещается решение системы.

```
PROCEDURE GAUS (N:INTEGER; A : MAS; B : MAS1;
                                VAR X : MAS1);
TYPE MST = ARRAY [1..N] OF REAL;
          MSS = ARRAY [1..N] OF MST;
VAR A1 : MSS; B1 : MST;
          I, J, M, K : INTEGER;   H : REAL;
BEGIN
     FOR I := 1 TO N DO
  BEGIN
          B1[I] := B[I];
    FOR J := 1 TO N DO
                    A1 [I,J] := A[I,J];
  END;
  FOR I := 1 TO N-1 DO
     FOR J := I+1 TO N DO
    BEGIN
                    A1[J,I] :=- A1[J,I] / A1[I,I];
      FOR K := I+1 TO N DO
                          A1[J,K] := A1 [J,K] + A1[J,I]*A1[I,K];
      B1[J] := B1[J] + B1[I]*A1[J,I];
    END;
      X[N] := B1[N] / A1[N,N];
  FOR I := N-1 DOWNTO 1 DO
  BEGIN
          H := B1[I];
          FOR J := I+1 TO N DO
                H := H - X[J]*A1[I,J];
          X[I] := H / A1[I,I];
     END;
END.
```

3. Метод простой итерации

Систему следует привести к виду $X=AX+F$. Строят последовательность векторов: X_0 – произвольный вектор; $X_1=AX_0+F$; $X_2=AX_1+F$; …; $X_n=AX_{n-1}+F$. Процесс сходится, если $\|A\|<1$ для какой-либо нормы матрицы.

Отдельные координаты вычисляют по формулам:

$$x_i^0 = f_i, \quad x_i^{(k)} = \sum_{j=1}^{n} a_{ij} x_j^{(k-1)} + f_i \quad (i = 1, 2, \ldots, n).$$

Точность вычислений можно оценить из следующего соотношения:

$$\left\|X^* - X^{(k)}\right\| \le \frac{\|A\|^k}{1-\|A\|} \cdot \|X_1 - X_0\|;$$

если $X_0=F$,

то $\left\| X^* - X^{(k)} \right\| \le \frac{\|A\|^{k+1}}{1-\|A\|} \cdot \|F\|,$

где X^* – точное решение.

Формальные параметры процедуры. Входные: ***A*** (тип ***real***) - матрица, составленная из коэффициентов при ***X*** преобразованного уравнения; ***B*** (тип ***real***) - матрица, составленная из свободных членов; *N* (тип ***integer***) - размерность матриц ***A*** (*N* × *N*) и ***B***(*N*)*; IK* (тип ***integer***) - предельно возможное количество итераций (введено для того, чтобы в случае расхождения процесса выйти из подпрограммы. Обычно решение достигается за 3-6 итераций; *EPS* (тип ***real***) - заданная погрешность решения. *Выходные:* ***X*** (тип ***real***) - матрица, в которой находится решение системы.

```
PROCEDURE ITER (N, IK :INTEGER; EPS : REAL;
                                A : MAS1; B : MAS; VAR X : MAS);
VAR  X1 : MAS;   S : REAL;    I, J, K : INTEGER;
BEGIN  X1 := B;
       X := X1;          K := 0;
  REPEAT                 S := 0.0;
              INC(K);
       FOR I := 1 TO N DO
              BEGIN
                     FOR J := 1 TO N DO X[I] := A[I,J]*X1[J] + B[J];
                     S := S + ABS (X[I]-X1[I]);
              END;
              S := S / N;                X1 := X;
       UNTIL (S<EPS) AND (K>IK);
    END.
```

4. Метод Зейделя

Систему следует привести к виду *X=AX+F*. Строят последовательность векторов $X_0, X_1, \ldots, X_k, \ldots$, где X_0 – произвольный вектор.

Координаты вектора X_k определяют по формуле:

$$x_i^{(k)} = \sum_{j=1}^{i-1} a_{ij} x_j^{(k)} + \sum_{j=1}^{n} a_{ij} x_j^{(k-1)} + f_i \quad (i = 1, 2, \ldots, n).$$

Условия сходимости и точности вычислений можно определить так же, как и в методе простой итерации.

Формальные параметры процедуры такие же, как и в п. 3. Для сравнения решим такую же систему линейных уравнений, как в методе итераций. Результат решения системы методом Зейделя (табл. 1).

Таблица 1

x[1]	*x*[2]	*x*[3]	*x*[4]	№ итерации
2.150000	0.440000	0.830000	1.160000	*ITER* = 0
1.252800	1.484800	1.229600	1.299200	*ITER* = 1
1.263936	1.523776	1.237952	1.315904	*ITER* = 2
1.265272	1.528453	1.238954	1.317908	*ITER* = 3
1.265433	1.529014	1.239075	1.318149	*ITER* = 4
1.265452	1.529082	1.239089	1.318178	*ITER* = 5
1.265452	1.529082	1.239089	1.318178 :	РЕШЕНИЕ

```
PROCEDURE ITER (N, IK :INTEGER; EPS : REAL;
                                A : MAS1; B : MAS; VAR X : MAS);
VAR  X1 : MAS;   S : REAL;    I, J, K : INTEGER;
BEGIN                   X1 := B;
        X := X1;
        K := 0;
  REPEAT                S := 0.0;
                INC(K);
        FOR I := 1 TO N DO
        BEGIN
                        FOR J := 1 TO N DO
                                X[I] := A[I,J]*X1[J] + B[J];
        S := S + ABS (X[I]-X1[I]);
                        X1 := X;
                END;
                S := S / N; X1 := X;
  UNTIL (S<EPS) AND (K>IK);
    END.
```

Лабораторная работа № 3. Интерполирование функций сплайнами.

1. Интерполяционная формула Лагранжа

$$P_n(x) = \sum_{i=0}^{n} y_i \frac{(x-x_0)(x-x_1)...(x-x_{i-1})(x-x_{i+1})...(x-x_n)}{(x_i-x_0)(x_i-x_1)...(x_i-x_{i-1})(x_i-x_{i+1})...(x_i-x_n)}.$$

При вычислении коэффициентов Лагранжа разности удобно расположить следующим образом (табл. 2).

Таблица 2

	$x_0 - x_1$	$x_0 - x_2$	……	$x_0 - x_n$
$x_1 - x_0$		$x_1 - x_2$	……	$x_1 - x_n$
$x_2 - x_0$	$x_2 - x_1$		……	$x_2 - x_n$
……	……	……	……	
$x_n - x_0$	$x_n - x_1$	$x_n - x_2$	……	

Если обозначить произведение элементов строк через D_i (i = 0, 1, …, n), а произведение элементов главной диагонали – через $\prod_{n+1}(x)$, то получится формула $P_n(x) = \prod_{n+1}(x)\sum_{i=0}^{n}\frac{y_i}{D_i}$.

В случае равноотстоящих узлов интерполяционная формула Лагранжа принимает вид $P_n(x) = \prod_{n+1}(t)\sum_{i=0}^{n}\frac{y_i}{(t-i)i!(n-i)!(-1)^{n-i}}$, где $t = (x - x_0)/h$, $h = x_{i+1} - x_i$ (i = 0, 1, 2, …, n).

Для оценки погрешности интерполяционной формулы Лагранжа можно использовать соотношение $|R_n(x)| \le \frac{M_{n+1}|\prod_{n+1}(x)|}{(n+1)!}$, где $M_{n+1} = \max_{[\alpha,b]}|f^{(n+1)}(x)|$.

2. Интерполяционные формулы Ньютона

а) *Первая интерполяционная формула Ньютона* имеет вид

$$P_n(x) = y_0 + q\Delta y_0 + \frac{q(q-1)}{2!}\Delta^2 y_0 + \ldots + \frac{q(q-1)\ldots(q-n+1)}{n!}\Delta^n y_0,$$

где $q = (x - x_0)/h$, $h = x_{i+1} - x_i$ ($i = 0, 1, \ldots, n$), $\Delta^i y_0$ – конечная разность *i–го* порядка, причем $\Delta^i y_0 = \Delta^{i-1} y_1 - \Delta^{i-1} y_0$ ($i = 1, 2, \ldots, n$).

Если $n = 1$, то получается *формула линейной интерполяции:*

$$P_1(x) = y_0 + q\Delta y_0.$$

Если $n = 2$, то получается *формула квадратичной интерполяции*:

$$P_2(x) = y_0 + q\Delta y_0 + \frac{q(q-1)}{2}\Delta^2 y_0.$$

б) *Вторая интерполяционная формула Ньютона* имеет вид

$$P_n(x) = y_n + q\Delta y_{n-1} + \frac{q(q+1)}{2!}\Delta^2 y_{n-2} + \ldots + \frac{q(q+1)\ldots(q+n-1)}{n!}\Delta^n y_0,$$

где $q = (x - x_n)/h$.

в) *Интерполяционная формула Ньютона для неравноотстоящих значений аргумента* имеет вид

$$P_n(x) = f(x_0) + (x - x_0) f(x_0, x_1) + (x - x_0)(x - x_1) f(x_0, x_1, x_2) + \ldots + (x - x_0)(x - x_1)(x - x_2)\ldots(x - x_{n-1}) f(x_0, x_1, \ldots, x_n),$$

где $f(x_0, x_1, \ldots, x_i) = \dfrac{f(x_1, x_2, \ldots, x_i) - f(x_0, x_1, \ldots, x_{i-1})}{x_i - x_0}$ - разделенная разность *i–го* порядка.

```
FUNCTION NEW1 (KEY:INTEGER;X,Y,R1,R2,R3,R4: MAS;
                              X1:REAL):REAL;
LABEL 30;
VAR I,J,K : INTEGER; Q : REAL;
BEGIN
      I := 1;
      J := 14;
      IF X1 < X[I] THEN
            GOTO 30;
                  { ****МЕТОД ДВОИЧНОГО ПОИСКА ИНТЕРВАЛА****}
      REPEAT
            K := (I+J) DIV 2;
            IF X1< X[K] THEN
                  J := K;
            IF X1>=X[K] THEN
                  I := K;
      UNTIL J <= I+1;
```

```
 30:
        Q := (X1 - X[I]) / (X[2]-X[1]);
                                { ****ВЫБОР ТИПА РАБОТЫ ПРОЦЕДУРЫ****}
 CASE KEY OF
                0:NEW1:=Y[I]+Q*(R1[I]+(Q-1)*(R2[I]/2.0+
                                (Q-2)*R3[I]/6.0));
                1: NEW1 := R1[I] + 0.5*(2*Q-1)*R2[I] +
                                ((3*Q-6)*Q+2)*R3[I]/6+

                                (((2*Q-9)*Q+11)*Q-3)*R4[I]/12;
                2: NEW1 := R2[I]+(Q-1)*R3[I]+
                                ((6*Q-8)*Q+11)*R4[I]/12;
        END;
 END.
                        {***ВТОРАЯ ИНТЕРПОЛЯЦИОННАЯ ФОРМУЛА НЬЮТОНА ***}
 FUNCTION NEW2 (KEY:INTEGER;X,Y,R1,R2,R3,R4:MAS;
                                X1:REAL):REAL;
 LABEL 30;
 VAR I,J,K : INTEGER; Q : REAL;
 BEGIN
        I := 1;
        J := 14;
        IF X1 > X[J] THEN GOTO 30;
 { **** МЕТОД ДВОИЧНОГО ПОИСКА ИНТЕРВАЛА ****}
        REPEAT
                K := (I+J) DIV 2;
                IF X1< X[K] THEN J := K;
                IF X1>=X[K] THEN I := K;
        UNTIL J <= I+1;
 30:
        Q := (X1 - X[I]) / (X[2]-X[1]);
                                { **** ВЫБОР ТИПА РАБОТЫ ПРОЦЕДУРЫ ****}
        CASE KEY OF
                 0: NEW2 := Y[J]+Q*(R1[J-1]+
                                        (Q+1)*(R2[J-2]/2.0+(Q+2)*R3[J-3]/6.0));
                1: NEW2 := R1[J-1]+0.5*(2*Q+
                                        1)*R2[J-2]+((3*Q+6)*Q+2)*R3[J-3]/6+
                                        (((2*Q+9)*Q+11)*Q+3)*R4[J-4]/12;
                 2: NEW2 := R2[J-2]+(Q+
                                    1)*R3[J-3]+((6*Q+18)*Q+11)*R4[J-4]/12;
        END;
END.
```

4. Интерполяционные формулы Гаусса

Первая формула Гаусса имеет вид

$$P(x) = y_0 + q\Delta y_0 + \frac{q(q-1)}{2!}\Delta^2 y_{-1} + \frac{(q+1)q(q-1)}{3!}\Delta^3 y_{-1} +$$
$$+ \frac{(q+1)q(q-1)(q-2)}{4!}\Delta^4 y_{-2} + \frac{(q+2)(q+1)q(q-1)(q-2)}{5!}\Delta^5 y_{-2} + \ldots +,$$
$$+ \frac{(q+n-1)\ldots(q-n+1)}{(2n-1)!}\Delta^{2n-1} y_{-(n-1)} + \frac{(q+n-1)\ldots(q-n)}{(2n)!}\Delta^{2n} y_{-n},$$

где $q = (x - x_0)/h$.

Лабораторная работа № 4. Приближенное вычисление определенных интегралов

Рассмотрим задачу вычисления определенного интеграла при помощи нескольких значений интегрируемой функции. Будем строить вычислительные правила следующего вида:

$$\int_a^b f(x)dx = \sum_{j=1}^{n} A_j f(x_j).$$

Эта формула называется формулой механических квадратур, $\sum_{j=1}^{n} A_j f(x_j)$ - квадратурной суммой, A_j - квадратурными коэффициентами, x_j - узлами или абсциссами квадратурного правила. Остаточным членом квадратурного правила называется величина $R_n = \int_a^b f(x)dx - \sum_{j=1}^{n} A_j f(x_j)$.

Возможны различные подходы к построению квадратурных формул.

Интерполяционные квадратурные формулы

Пусть заданы значения подынтегральной функции f(x) в точках x_0, x_1, …, x_n принадлежащих [a,b], тогда для f(x) строят интерполяционный многочлен Лагранжа n-ой степени, т.е.

$$\int_a^b f(x)dx = \int_a^b L_n(x)dx = \sum_{j=0}^{n} A_j f(x_j), \text{ где } A_j = \int_a^b \frac{\omega_n(x)}{(x-x_j)\omega_n'(x_j)}dx.$$

Эта формула называется интерполяционной квадратурной формулой. Её остаточный член имеет вид

$$R_n = \int_a^b r(x)dx = \frac{f^{(n+1)}(\eta)}{(n+1)!}\int_a^b \frac{\omega_n(x)}{(x-x_j)\omega_n'(x_j)}dx,$$ где η - некоторая точка [a,b].

Если узлы квадратурного правила равноотстоящие, то квадратурные коэффициенты принимают вид

$$A_j = \frac{h(-1)^{n-j}}{j!(n-j)!}\int_a^b \frac{t(t-1)\ldots(t-n)}{(t-j)}dt, \text{ где } t = \frac{x-x_0}{h},\ x_i - x_{i-1} = h,\ i = \overline{0,n-1}.$$

Формальные параметры процедуры. Входные: *X*0, *X*1 (тип ***real***) - границы отрезка, на котором вычисляется определенный интеграл; *N* (тип ***integer***) - количество узлов (элементарных отрезков), на которые разбивается заданный интервал [*a, b*]; *к* (тип ***integer***) - параметр, определяющий тип расчета (*к*=1 - формула левых прямоугольников; *к*=2 - формула правых прямоугольников; *к*=3 - формула средних прямоугольников; *к*=4 - формула трапеций). *Выходные*: *FX* (тип ***real***) - массив значений функции в заданных узлах (x_i); *fint* (тип ***real***) - вычисленные значения интеграла; *err* (тип ***real***) - погрешность вычисленного значения.

```
PROCEDURE INTLPS (VAR S:REAL; K:INTEGER);
BEGIN
        S := 0.0;
        X := A;
        FOR I := 0 TO N DO
        BEGIN
                IF K <>3 THEN X := A+ H*I
                        ELSE X:=A+H/2+H*I;
        F := FUNC (A,B,X);
                IF K=4 THEN F := F/2.0;
                IF (I=0) AND (K<>2) THEN S := S + F;
                IF (I=N) AND ((K=2) OR (K=4)) THEN S := S + F;
                IF (I<>0) AND (I<>N) THEN
                        IF K<>4 THEN S := S+F
                                ELSE S := S+F*2;
        END;    S := S * H;
 END.
```

Квадратурные формулы Ньютона-Котеса.

Запишем квадратурное правило для равноотстоящих узлов в виде

$$\int_a^b f(x)dx \cong (b-a)\sum_{j=0}^{n} C_j^n f(x_j), \text{ где } C_j^n = \frac{A_j}{(b-a)} = \frac{(-1)^{n-j}}{n\,j!(n-j)!}\int_0^n \frac{t(t-1)\ldots(t-n)}{t-j}dt.$$

При заданных значениях n коэффициенты принимают следующие значения:

$$n=0,\ C_0^0 = 1;\ n=1,\ C_0^1 = C_1^1 = \frac{1}{2};$$

$$n=2,\ C_0^2 = C_2^2 = \frac{1}{6},\ C_1^2 = \frac{4}{6};\ n=3,\ C_0^3 = C_3^3 = \frac{1}{8},\ C_1^3 = C_2^3 = \frac{3}{8} \text{ и т.д.}$$

Формула прямоугольников

Пусть функция f(x) на [a,b] заменяется интерполяционным многочленом Лагранжа нулевого порядка, построенным по значению в средней точке отрезка[a,b], т.е. $x = \frac{a+b}{2}$ и $L_0(x) = f\left(\frac{a+b}{2}\right)$. Тогда

$$\int_a^b f(x)dx = (b-a)f\left(\frac{a+b}{2}\right) + R_0 \text{ и } R_0 = \frac{(b-a)^3}{24} f''(\xi), \text{ где } \xi \in [a,b].$$

Разделим [a,b] на m равных частей длины $h = \frac{b-a}{m}$. К каждому частичному отрезку [a+ih,a+(i+1)h] применим формулу прямоугольников, сложив результаты, получим обобщенную формулу прямоугольников

$$\int_a^b f(x)dx = \frac{b-a}{m}\left[f\left(a+\frac{h}{2}\right) + f\left(a+\frac{3h}{2}\right) + \ldots + f\left(a+\frac{(2m-1)h}{2}\right)\right] + \frac{(b-a)^3}{24m^2} f''(\xi).$$

Формула трапеций

Пусть функция f(x) на [a,b] заменяется интерполяционным многочленом первого порядка, построенным по значениям в точках a и b. Тогда

$$\int_a^b f(x)dx = \frac{(b-a)}{2}\left[f(a) + f(b)\right] + R_1 \text{ и } R_1 = -\frac{(b-a)^3}{12} f''(\xi), \text{ где } \xi \in [a,b].$$

Разделим [a,b] на m равных частей длины $h = \frac{b-a}{m}$. К каждому частичному отрезку [a+ih,a+(i+1)h] применим формулу трапеций, сложив результаты и обозначив $f(a+ih) = f_i$, получим обобщенную формулу трапеций

$$\int_a^b f(x)dx = \frac{b-a}{m}\left[\frac{f_0 + f_m}{2} + f_1 \ldots + f_{m-1}\right] - \frac{(b-a)^3}{12m^2} f''(\xi).$$

Формула Симпсона (формула парабол)

Пусть функция f(x) на [a,b] заменяется интерполяционным многочленом второго порядка, построенным по значениям в точках a, $\frac{a+b}{2}$ и b. Тогда

$$\int_a^b f(x)dx = \frac{(b-a)}{6}\left[f(a) + 4f\left(\frac{a-b}{2}\right) + f(b)\right] + R_2 \text{ и } R_1 = -\frac{1}{90}\left(\frac{b-a}{2}\right)^5 f^{(IV)}(\xi), \text{ где } \xi \in [a,b].$$

Разделим [a,b] на четное число m равных частей длины $h = \frac{b-a}{m}$. К каждому частичному отрезку [a+(i-1)h,a+(i+1)h] применим формулу Симпсона, сложив результаты и обозначив $f(a+ih) = f_i$, получим обобщенную формулу Симпсона

$$\int_a^b f(x)dx = \frac{b-a}{3m}\left[f_0 + f_m + 2(f_2 + f_4 + \ldots + f_{m-2}) + 4(f_1 + f_3 + \ldots + f_{m-1})\right] - \frac{(b-a)^5}{180m^4} f^{(IV)}(\xi).$$

```
PROCEDURE INTLPS (VAR S:REAL; K:INTEGER);
BEGIN
    S := 0.0;
    X := A;
    IF K<5 THEN
        FOR I := 0 TO N DO
        BEGIN
            IF K <>3 THEN X := A+ H*I
                ELSE X:=A+H/2+H*I;
            F := FUNC (X);
            IF K=4 THEN F := F/2.0;
            IF (I=0) AND (K<>2) THEN S := S + F;
            IF (I=N) AND ((K=2) OR (K=4)) THEN S := S + F;
            IF (I<>0) AND (I<>N) THEN
            IF K<>4 THEN S := S+F
                ELSE S := S + F*2;
        END
    ELSE
    BEGIN
        FOR I := 1 TO N-1 DO
            S := 2*FUNC(X+H*I) + 4*FUNC(X+H*I-H/2) + S;
        S := S + 4*FUNC(B-H/2) + FUNC(A) + FUNC(B);
        S := S / 6;
    END;
    S := S * H;
END.
```

Лабораторная работа № 5. Приближенное вычисление определенных интегралов.

Формулы численного интегрирования

а) Пусть отрезок интегрирования [*a*, *b*] разбит на *n* частей с шагом *h=(b–a)/n*.

Тогда $\int_a^b f(x)dx \approx h\sum_{i=0}^{n-1} f(x_i)$ *(формула левых прямоугольников);*

$\int_a^b f(x)dx \approx h\sum_{i=1}^{n} f(x_i)$ *(формула правых прямоугольников);*

$\int_a^b f(x)dx \approx h\sum_{i=0}^{n-1} f(x_i + \frac{h}{2})$ *(формула средних прямоугольников),*

где $x_i = a + ih$ *(i*= 0, 1, 2,…,*n).*

Остаточные члены этих формул соответственно равны

$$R_n(f) = \frac{(b-a)^2}{2n} f'(\varepsilon);$$

$$R_n(f) = -\frac{(b-a)^2}{2n} f'(\varepsilon);$$

$$R_n(f) = \frac{(b-a)^3}{24n^2} f''(\varepsilon), \quad \text{где } a \le \varepsilon \le b.$$

б) *Формула Ньютона – Котеса*

$$\int_a^b f(x)dx \approx (b-a)\sum_{k=0}^{n} C_{kn} f(x_k),$$

где $x_i = a + ih; (i = 0,1,2,\ldots,n), x_0 = a, h = (b-a)/n.$

Коэффициенты $C_k = H_{kn}/N_n$ определены заранее и могут быть взяты из таблицы (табл.3).

Таблица 3

n	H_{0n}	H_{1n}	H_{2n}	H_{3n}	H_{4n}	H_{5n}	H_{6n}	N_n
1	1	1						2
2	1	4	1					6
3	1	3	3	1				8
4	7	32	12	32	7			90
5	19	75	50	50	75	19		288
6	41	216	27	272	27	216	41	840

```
   FUNCTION NEWCOT (A,B:REAL;N:INTEGER) : REAL;
 TYPE MS = ARRAY [0..7] OF INTEGER;
              MS2 = ARRAY [3..7] OF MS;
              FF = FILE OF MS2;
 VAR H1,H2 : MS2; FC : TEXT;
              X1, SUM,H, Y1 : REAL;
 BEGIN
       ASSIGN (FC,'A:\');
       RESET (FC);
       FOR I := 3 TO 7 DO
              FOR J := 0 TO 7 DO
                     READ (FC,H1[I,J]);
       FOR I := 3 TO 7 DO
              FOR J := 0 TO 7 DO
                     READ (FC,H2[I,J]);
       CLOSE (FC);
       H := (B-A) / N;
       X1 := A;
       SUM := 0.0;
       FOR I := 0 TO 7 DO
       BEGIN
              SUM := SUM + FUNC (X1) * H1[N,I]/H2[N,I];
              X1 := X1 + H;
       END;
       NEWCOT := (B-A) * SUM;
END
```

в) *Формула трапеций* иммет вид

$$\int_a^b f(x)dx \approx h\left(\frac{1}{2}y_0 + y_1 + y_2 + \ldots + y_{n-1} + \frac{1}{2}y_n\right),$$

где $y_i = f(x_i), (i = 0,1,2,\ldots,n)$, причём $R_n(f) = -\frac{(b-a)^3}{24n^2} f''(\varepsilon)$, $a \le \varepsilon \le b$.

г) *Формула Симпсона (число n – обязательно чётное)*

$$\int_a^b f(x)dx \approx \frac{h}{3}[y_0 + y_n + 2(y_2 + y_n + y_6 + ... + y_{n-2}) + 4(y_1 + y_3 + ... + y_{n-1})],$$

причём $R_n(f) = -\frac{(b-a)^5}{180n^4} f^{|V}(\varepsilon)$, $a \le \varepsilon \le b$.

д) *Формула Гаусса*

$$\int_a^b f(x)dx \approx \frac{b-a}{2}[C_1 f(x_1) + C_2 f(x_2) + ... + C_n f(x_n)],$$

где $x_i = \frac{b+a}{2} + \frac{b-a}{2} t_i, (i = 1,2,...,n)$.

Значения t_i и C_i берутся из таблицы (табл.4)

Таблица 4

n	t_i	C_i
1	$t_1 = 0$	$C_1 = 2{,}000000$
2	$t_{1,2} = \pm 0{,}577350$	$C_1 = C_2 = 1{,}000000$
3	$t_{1,3} = \pm 0{,}774597$ $t_2 = 0$	$C_1 = C_3 = 0{,}555556$ $C_2 = 0{,}888889$
4	$t_{1,4} = \pm 0{,}861136$ $t_{2,3} = \pm 0{,}339981$	$C_1 = C_4 = 0{,}347855$ $C_2 = C_3 = 0{,}652145$

.

Лабораторная работа № 6. Метод Рунге-Кутта и Эйлера

Решение задачи Коши для дифференциальных уравнений n-го порядка

Задача Коши для дифференциального уравнения *n*-го порядка $y^{(n)} = f(x,y,y',y'',...,y^{(n-1)})$ заключается в отыскании функции y(x), удовлетворяющей этому уравнению и начальным условиям $y(x_0) = y_0, y'(x_0) = y'_0, ..., y^{(n-1)}(x_0) = y_0^{(n-1)}$, где $x_0, y_0, y'_0, ..., y_0^{(n-1)}$ – заданные числа.

Для решения поставленной задачи можно применить один из следующих методы, рассмотренных ниже.

Метод Эйлера

Численное решение задачи состоит в построении таблицы приближенных значений y_1, y_2, ..., y_n, являющихся решениями уравнения y = y(x) в точках x_1, x_2, ..., x_n. Чаще всего точки x_i расположены равномерно: $x_i = x_0 + h \cdot i$, где i = =1, 2, ..., n. Точки x_i называют узлами сетки, h - шагом сетки. Ясно, что h > 0.

Для получения решения задачи Коши воспользуемся разложением функции в ряд Тейлора. Для этого продифференцируем исходное уравнение y' = f(x,y) по x n раз. Тогда с учетом правила дифференцирования сложной функции получим следующие соотношения:

$y'' = f_x(x, y) + f_x(x, y) \cdot y';$

$y''' = f_{xx}(x,y) + 2f_{xy}(x,y) \cdot y' + f_{yy}(x,y) \cdot (y') + f_y(x,y) \cdot y'';$ . . .

Полагая $x = x_0$ и $y = y_0$, получаем ряд $y'(x_0); y''(x_0); y'''(x_0); ... ; y^{(n)}(x_0)$, который сходится к решению поставленной задачи, т.е.

$$y(x) \cong \sum_{i=0}^{n} y^{(i)}(x_0) \cdot \frac{(x - x_0)^i}{i!}.$$

Надо заметить, что если $|x - x_0|$ больше радиуса сходимости ряда $y'(x_0); y''(x_0); y'''(x_0); ...; y^{(n)}(x_0)$, то погрешность $| y(x_0) - y|$ не стремится к нулю при $n \to \infty$, т.е. ряд расходится. Тогда поступают следующим образом. Отрезок $[x_0, x_0+X]$, на котором ряд расходится, разбивают на меньшие отрезки $[x_i, x_{i+1}]$, где i = 1, 2, ..., n, и получают новые последовательные приближения y_j к решению $y(x_j)$ при j = 1, 2, ..., m, используя следующую схему:

1) y_i считаем найденным (для первого шага им может быть y_0);
2) вычисляем в точке x_i производные $y_i^{(k)}$;
3) полагаем $y_x \cong z_i(x) \cong \sum_{k=0}^{n} y^{(k)}(x_0) \cdot \frac{(x - x_0)^k}{k!}$;
4) считаем $y_{i+1} = z_i(x_{i+1})$;
5) повторяем с первого пункта.

Если же в формуле положить n=1, то получим основную расчетную формулу метода Эйлера: $y_{i+1}=y_i+h\times\times f(x,y)$.

Для оценки погрешности метода Эйлера на одном из шагов сетки разложим точное решение в ряд Тейлора в окрестности узла x_i:

$$y(x_{i+1}) = y(x_{i+h}) = y(x_i) + y'(x_i)\cdot h + o(h^2) = y(x_i) + f(x_i,y_i)\cdot h + o(h^2).$$

Сравнение разложения с основной расчетной формулой метода показывает, что они совпадают до членов первого порядка по h, а погрешность формулы равна $o(h^2)$, т.е. метод Эйлера относится к методам первого порядка.

Программа, реализующая метод Эйлера, может быть представлена простой процедурой-функцией

```
function eyler (x, h, y : real) : real;
begin
eyler := y + h*FUNC(x,y)
end.
```

Формальные параметры процедуры. Входные: x (тип real) - очередное значение x_i; h (тип real) - величина шага разбиения отрезка, на котором ищется решение; y (тип real) - предыдущее значение y_i; FUNC(x) (тип real) - процедура-функция, вычисляющая значение правой части уравнения по заданным x и y. Результатом работы данной процедуры-функции явится очередное значение y_{i+1}.

Для проверки и тестирования процедуры решалась задача Коши методом Эйлера на отрезке [0.2; 1.2] с шагом 0.1 при начальном условии y(0.2) = 0.25. Вычисления выполнялись с точностью 10^{-4}:

$$y' = 0.185(x^2 + \cos(0.7x) + 1.843y.$$

Составим процедуру-функцию для вычисления правой части уравнения:

```
function FUNC (x, y : real) : real;
begin
    func := 1.843*y + 0.185*(x*x + Cos(0.7*x));
end.
```

Результаты работы программы приведены в табл. 5.

Таблица 5

i	x_i	y_i
1	0.200000	0.250000
2	0.300000	0.315134
3	0.400000	0.392972
4	0.500000	0.486136
5	0.600000	0.597734
6	0.700000	0.731449
7	0.800000	0.891643
8	0.900000	1.083487

9	1.000000	1.313107
10	1.100000	1.587762
11	1.200000	1.916053

Метод Рунге-Кутта

Методы Рунге - Кутта относятся к многошаговым методам повышенной точности (от второго порядка и выше). Здесь будет рассмотрен одношаговый метод Рунге - Кутта четвертого порядка точности.

Пусть требуется найти решение дифференциального уравнения y' = f(x, y), удовлетворяющего начальному условию $y(x_0) = y_0$.

Численное решение задачи состоит в построении таблицы приближенных значений y_0, y_1, ..., y_n решения уравнения y' = f(x,y) в точках x_0, x_1, ..., x_n, которые называют узлами сетки. Для краткости обозначим $x_i = x_0 + ih$; $y_i = y(x_i)$, где h - шаг сетки (ясно, что h > 0).

Обычно методом Рунге-Кутта в литературе называют метод, согласно которому у искомой функции вычисляют по формулам:

$$y_{i+1} = y_i + k/6,$$

где обозначено:

$k = k_1 + 2k_2 + 2k_3 + k_4$;

$k_1 = hf(x_i, y_i)$;

$k_2 = h\, f(x_i + h/2 ,y_i + k_1/2)$;

$k_3 = h\, f(x_i + h/2 ,y_i + k_2/2)$;

$k_4 = h\, f(x_i + h ,y_i + hk_3)$.

Погрешность метода. Если $y_i^{(h)}$ - приближение, вычисленное с шагом h, а $y_i^{(h/2)}$ - с шагом h/2, то справедлива оценка

$$\left| y_i^{(h/2)} - y(x_i) \right| \le \frac{16}{5} \cdot \left| y_i^{(h/2)} - y_i^{(h)} \right|.$$

Формальные параметры процедуры. Входные: N (тип integer) - количество разбиений отрезка [a, b]; X, Y (тип real) - начальные значения, соответствующие x_0 и y_0; h (тип real) - шаг интегрирования системы; FUNC - имя процедуры-функции, по которой вычисляют значения правой части уравнения y' = f(x, y). Выходные: YR (тип real) - массив, содержащий решения системы в точках $x_i = x_0 + hi$.

```
PROCEDURE RGK (N:INTEGER; Y,H,X : REAL;  VAR YR : MAS);
VAR I : INTEGER; F1, F, Y0, X0 : REAL;
BEGIN  I := 2;   Y0 := Y;
       X0 := X;
       YR[1] := Y0;
       REPEAT
              F1 := H * FUNC (X0,Y0);
```

```
            F := H * FUNC (X0+H/2,Y0+F1/2);
            F1 := F1 + F*2;
            F := H * FUNC (X0+H/2,Y0+F/2);
            F1 := F1 + F*2;
            F := H * FUNC (X0+H,Y0+F);
            F1 := F1 + F;
            YR[I]:=YR[I-1] + F1 / 6;
            Y0 := YR [I];
            X0 := X0 + H;
            INC (I);
        UNTIL I>N;
END;
```