МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Казанский национальный исследовательский технический университет им. А.Н. Туполева-КАИ»
(КНИТУ-КАИ)

Институт Компьютерных технологий и защиты информации
(наименование института)
Кафедра Прикладной математики и информатики
(наименование кафедры)

ОТЧЕТ
по лабораторной работе 4
Интерполирование функций

Номер индивидуального задания: 7

Выполнил обучающийся группы 4312
Д.Д.Наумихин
(ФИО)

Казань 2024

1 Цель работы

Построение интерполяционных многочленов, многочленов для численного дифференцирования по заданной системе точек, используя ЭВМ.

2 Содержание работы

1. Построение интерполяционной формулы Лагранжа, первой и второй интерполяционной формул Ньютона и апроксимационнного полинома;
2. На заданном примере построение полиномы на ЭВМ;
3. Составление программы на любом языке программирования (выбран ЯП семейства Lisp, диалекта Common Lisp), реализующую процесс построения указанных полиномов второго порядка для системы из равноотстоящих узловых точек;
4. Анализ точности построения полиномов;
5. Составление отчета о проделанной работе.

3 Задание

3.1 Ход работы

1. Составить таблицу значений экспериментальной функции $y = 18\sin(\sqrt{x^3}+8)$ для равноотстоящей системы из шести узловых точек $x_{i+1} = x_i + h, i = \overline{0,\ 5}$ на отрезке из области допустимых значений функции, где h = 5;
2. По сформированной системе точек построить интерполяционные формулы Лагранжа, I и II интерполяционные формулы Ньютона;
3. Составить программу ена любом языке программирования (выбран язык Common Lisp), реализующую процесс построения указанных полиномов для заданной системы точек.

4 Вводные данные в примере

4.1 Таблица значений

Для данного примера таблица значений

$$y_i = f(x_i),\ x_i = x_0 + ih = x_0 + i \cdot \frac{x_n - x_0}{n}, i = \overline{0, n} \tag{1}$$

$$\begin{cases} x_0 = 3.36 \\ x_n = 3.37 \\ n = 5 \end{cases} \tag{2}$$

Принимает вид:

x_i	3.35	$3.351(6)$	$3.35(3)$	3.355	$3.35(6)$	$3.358(3)$	3.36
y_i	17.9997116	17.999989	17.9998901	17.999413	17.998559	17.997327	17.9957172

4.2 Таблица разностей

x_0	x_1	x_2	x_3	x_4	x_5
2.777662935606884d-4	-9.927312368773755d-5	-4.765917023163979d-4	-8.541815035840727d-4	-0.0012320345693339618d0	-0.0016101429221535568d0
-3.7703941724842593d-4	-3.7731857862866036d-4	-3.7758980126767483d-4	-3.778530657498891d-4	-3.78108352819595d-4	
-2.791613731289999d-7	-2.7122264256718154d-7	-2.632644857669675d-7	-2.552870590477596d-7		
7.938734114532053d-9	7.958160352927734d-9	7.97742316649419d-9			
1.9408474827287137d-11	1.9259260852777516d-11				
-1.9539925233402755d-13					

5 Решение

5.1 I интерполяционная формула Ньютона

5.1.1 Теоретический вывод

Задача такова: необходимо с помощью набора простых функций (можно назвать базисом), состоящим из функций

$$\{e_i\} = \{1\} \cup \left\{ \prod_{i=0}^{k} (x - x_i) \colon k = \overline{0, n} \right\} \tag{3}$$

Из равноудаленности точек, данные базисные функции можно записать кратко:

$$x^{[k]} = \begin{cases} \prod_{i=0}^{k-1} (x - ih), & k > 0 \\ 1, & k = 0 \end{cases}, \; k = \overline{0, n} \tag{4}$$

$$\{e_i\} = \left\{ (x - x_0)^{[k]} \colon k = \overline{0, n} \right\} \tag{5}$$

Далее мы должны найти такую их линейную комбинацию, что в заданных равностоящих узлах. Этот полином принимает целевые значения:

$$P_n(x_i) = f(x_i) = y_i \tag{6}$$

Шаблон полинома:

$$P(x) = \sum_{k=0}^{n} a_k (x - x_0)^{[k]} \tag{7}$$

При последовательной подстановке точек x_i в определение ф-л. (6) можно найти искомые коэффициенты a_k:

$$x = x_i \Longrightarrow x - x_i = 0 \tag{8}$$

$$\forall i, j = \overline{0, n} \colon \; j > i \Longrightarrow \prod_{k=0}^{j-1} (x - x_k) = (x - x_0)^{[j]} = 0 \tag{9}$$

Аналогично:

$$x = x_i \tag{10}$$

$$\forall i, j = \overline{0, n} \colon \; j \le i \Longrightarrow \prod_{k=0}^{j-1} (x - x_k) = (x_i - x_0)^{[j]} = (x_0 + ih - x_0)^{[j]} = (ih)^{[j]} \tag{11}$$

$$(ih)^{[j]} = \prod_{k=0}^{j-1} (ih - kh) = \prod_{k=0}^{j-1} h(i - k) = h^j \frac{i!}{(i-j)!} = h^j \lambda_{ij} \tag{12}$$

Итак, формируются коэффициенты вида:

$$b_{ij} = \begin{cases} h^j \lambda_{ij}, & j \le i \\ 0, & j > i \end{cases} \tag{13}$$

$$\lambda_{ij} = \frac{i!}{(i-j)!} \tag{14}$$

В итоге получаем матричное уравнение:

$$\begin{pmatrix} \frac{0!}{(0-0)!} & 0 & \cdots & 0 \\ \frac{1!}{(1-0)!} & \frac{1!}{(1-1)!} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \frac{n!}{(n-0)!} & \frac{n!}{(n-1)!} & \cdots & \frac{n!}{(n-n)!} \end{pmatrix} \operatorname{diag}\{h^k \colon k = \overline{0,n}\} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{pmatrix} \tag{15}$$

Матрица получилась треугольная. Необходимо найти решение для a_k. Для нахождения можно вычитать одну строку из другой, умножать на определенное число. Вычитание одной строки из другой (соседней) означает, что мы находим:

$$\Delta P(x_k) = P(x_k + h) - P(x_k) = P(x_{k+1}) - P(x_k) \tag{16}$$

При том вычитаются и решения:

$$\Delta f(x_k) = f(x_k + h) - f(x_k) = f(x_{k+1}) - f(x_k) = y_{k+1} - y_k \tag{17}$$

Коэффициенты здесь таковы, что:

$$\begin{pmatrix} 1 & k & k(k-1) & \cdots & k! & 0 & \cdots & 0 \\ 1 & (k+1) & (k+1)k & \cdots & (k+1)! & (k+1)! & \cdots & 0 \end{pmatrix} \tag{18}$$

Вычитая строки:

$$\begin{pmatrix} 0 & 1 & 2k & \cdots & k!k & k! & \cdots & 0 \end{pmatrix} \tag{19}$$

В общем случае:

$$\begin{aligned} \lambda_{(k+1)t} - \lambda_{kt} &= \frac{(k+1)!}{(k+1-t)!} - \frac{k!}{(k-t)!} = \frac{(k+1)!}{(k+1-t)!} - \frac{((k+1)-t)}{((k+1)-t)}\frac{k!}{(k-t)!} = \\ &= \frac{(k+1)k!}{(k+1-t)!} - \frac{(k+1-t)k!}{(k+1-t)!} = \frac{k!(k+1-k-1+t)}{(k+1-t)!} = t\frac{k!}{(k+1-t)!} = \\ &= t\frac{k!}{(k+1-t)!} = t\frac{k!}{(k-(t-1))!} = t\lambda_{k(t-1)} \end{aligned} \tag{20}$$

То есть, мы имеем преобразование следуюещго плана:

$$\delta \colon \lambda_{kt} \mapsto \begin{cases} t\lambda_{k(t-1)}, & t > 0 \\ 0, & t = 0 \end{cases} \tag{21}$$

Повторное применение преобразования:

$$\delta^2 \lambda_{kt} = \delta\delta\lambda_{kt} = \delta t\lambda_{k(t-1)} = t\delta\lambda_{k(t-1)} = t(t-1)\lambda_{k(t-2)} \tag{22}$$

Следующее утверждение — результат рассуждений:

$$\delta^m \lambda_{kt} = \frac{t!}{(t-m)!}\lambda_{k(t-m)} \tag{23}$$

Proof. Доказывается по индукции:

1. База доказана в ф-л. (20).

2. Переход:

$$\delta^{m+1}\lambda_k t = \delta\delta^m\lambda_k t = \frac{t!}{(t-m)!}\delta\lambda_{k(t-m)} = \\ = \frac{t!}{(t-m)!}(t-m)\lambda_{k(t-(m+1))} = \frac{t!}{(t-(m+1))!}\lambda_{k(t-(m+1))} \quad (24)$$

□

Из определения коэффициентов ф-л. (14) видно, что:

$$\lambda_{k0} = \frac{k!}{k!} = 1 \quad (25)$$

Значит:

$$\begin{pmatrix}1 & k & k(k-1) & \cdots & k! & 0 & \cdots & 0\end{pmatrix} \overset{\delta^k}{\longmapsto} \begin{pmatrix}0 & 0 & 0 & \cdots & k! & \cdots & 0\end{pmatrix} \quad (26)$$

Далее:

$$\begin{pmatrix}1 & 0 & 0 & \cdots & 0 \\ 0 & h & 0 & \cdots & 0 \\ 0 & 0 & 2h^2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & n!h^n\end{pmatrix} \begin{pmatrix}a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_n\end{pmatrix} = \begin{pmatrix}\Delta^0 y_0 \\ \Delta^1 y_0 \\ \Delta^2 y_0 \\ \vdots \\ \Delta^n y_0\end{pmatrix} \quad (27)$$

$$\begin{pmatrix}a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_n\end{pmatrix} = \begin{pmatrix}y_0 \\ \frac{\Delta y_0}{h} \\ \frac{\Delta^2 y_0}{2h^2} \\ \vdots \\ \frac{\Delta^n y_0}{n!h^n}\end{pmatrix} \quad (28)$$

Коэффициенты найдены. Формула принимает следующий вид:

$$P_n(x) = \sum_{k=0}^{n} \frac{\Delta^k y_0}{k!h^k}(x-x_0)^{[k]} \quad (29)$$

Возможна замена:

$$q = \frac{x-x_0}{h} \implies \frac{x-x_k}{h} = \frac{x-x_0-kh}{h} = \frac{x-x_0}{h} - k = q-k \quad (30)$$

$$P_n(x) = \sum_{k=0}^{n} \frac{\Delta^k y_0}{k!} \prod_{m=0}^{k-1}(q-m) \quad (31)$$

5.1.2 Формула для заданного примера

$$P = 17.999711660684547 + 2.777662935606884\, q10^{-4} + -3.7703941724842593\, q(q-1)10^{-4} + \\ + -2.791613731289999\, q(q-1)(q-2)10^{-7} + 7.938734114532053q(q-1)(q-2)(q-3)10^{-9} + \\ + 1.9408474827287137\, q(q-1)(q-2)(q-3)(q-4)10^{-11} + \\ + -1.9539925233402755\, q(q-1)(q-2)(q-3)(q-4)(q-5)10^{-13} \quad (32)$$

```lisp
(in-package #:num-methods.lab1)

(defun first-newthon.q (nde x)
  (/ (- x (nde-beg nde))
    (nde-h nde)))

(defun first-newthon (nde f x)
  (let ((n (nde-n nde))
        (q (first-newthon.q nde x))
        (x_0 (nde-beg nde)))
    (labels ((first-newthon_ (k qnt_k sum_k)
               (if (> k n) sum_k
                   (let* ((k1 (1+ k))
                          (qnt_k1 (* qnt_k (/ (- q k)
                                             k1)))
                          (sum_k1 (+ sum_k (* qnt_k (finite-subs f x_0 k nde)))))
                    (first-newthon_ k1 qnt_k1 sum_k1)))))
      (first-newthon_ 0 1 0))))

```

```lisp
(in-package #:num-methods.lab1)

(defun second-newthon.p (nde x)
  (/ (- x (nde-end nde))
     (nde-h nde)))

(defun second-newthon (nde f x)
  (let ((n (nde-n nde))
        (p (second-newthon.p nde x)))
    (labels ((second-newthon_ (k qnt_k sum_k)
              (if (> k n) sum_k
                  (let* ((k1 (1+ k))
                         (qnt_k1 (* qnt_k (/ (+ p k)
                                            k1)))
                         (sum_k1 (+ sum_k (* qnt_k
                                             (finite-subs
                                               f (nde-i nde (- n k))
                                               k nde)))))
                    (second-newthon_ k1 qnt_k1 sum_k1)))))
      (second-newthon_ 0 1 0))))

```

```lisp
(in-package #:num-methods.lab1)

(defun lagrange.q (nde x)
  (/ (- x (nde-beg nde))
     (nde-h nde)))

(defun lagrange.c_i (nde f i)
  (/ (funcall f (nde-i nde i))
```

```
     (reduce #'*
             (mapcar
               #'(lambda (j) (progn
                               (if (= j i) 1 (- (nde-i nde i)
                                                (nde-i nde j)))))
               (iota (1+ (nde-n nde)))))))

(defun lagrange.c (nde f)
  (let ((n (nde-n nde)))
    (loop for i from 0 to n
          collect (lagrange.c_i nde f i))))

(defun lagrange (nde f x)
  (let ((n (nde-n nde))
        (q (lagrange.q nde x)))
    (labels ((lag_prod (i j res)
               (if (> j n) res
                   (let* ((j1 (1+ j))
                          (res1 (if (= i j) res
                                    (* res (/ (- q j)
                                            (- i j))))))
                     (lag_prod i j1 res1))))
             (lag_sum (i res)
               (if (> i n) res
                   (let* ((i1 (1+ i))
                          (res1 (+ res (* (funcall f (nde-i nde i))
                                          (lag_prod i 0 1)))))
                     (lag_sum i1 res1)))))
      (lag_sum 0 0))))

(defun lagrange-2 (nde f x)
  (let* ((n (nde-n nde))
         (l_ (curry #'lagrange.c_i nde f))
         (x_ (curry #'nde-i nde)))
    (reduce #'+
            (loop for i from 0 to n collect
                  (* (funcall l_ i)
                     (reduce
                       #'*
                       (loop for j from 0 to n collect
                             (if (= i j) 1
                                 (- x (funcall x_ j))))))))))
```