МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра математического обеспечения и применения ЭВМ

ОТЧЕТ

по практической работе №5

по дисциплине «Вычислительная математика»

ТЕМА: АЛГОРИТМ РЕМЕЗА

Студент гр. 0303 _____ Калмак Д.А.

Преподаватель _____ Сучков А.И.

Санкт-Петербург

2021

Цель работы.

Освоение и реализация алгоритма Ремеза для построения полиномов наилучшего равномерного приближения средствами GNU Octave.

Основные теоретические положения.

Алгоритм Ремеза (алгоритм замены Ремеза) – это итеративный алгоритм равномерного аппроксимирования функций $f \in C[a,b]$, основанный на теореме П. Л. Чебышёва об альтернансе. Предложен Е. Я. Ремезом в 1934 году.

Теоретической основой алгоритма Ремеза является следующая теорема:

Теорема. Для того, чтобы некоторый многочлен $P^*(x)$ степени не выше n был многочленом, наименее уклоняющимся от $f \in C[a,b]$, необходимо и достаточно, чтобы на $[a,b]$ нашлась по крайней мере одна система из $n+2$ точек x_i, $a \leqslant x_0 < x_1 < \cdots < x_{n+1} \leqslant b$, в которых разность $f(x) - P^*(x)$:

1. поочерёдно принимает значения разных знаков,
2. достигает по модулю наибольшего на $[a,b]$ значения.

Такая система точек называется чебышёвским альтернансом.

Пусть E_n – величина наилучшего приближения функции $f(x)$ многочленами степени n. Оценку E_n снизу даёт следующая теорема:

Теорема Валле-Пуссена. Если для функции $f \in C[a,b]$ некоторый многочлен $P(x)$ степени n обладает тем свойством, что разность $f(x) - P(x)$ на некоторой системе из $n+2$ упорядоченных точек x_i принимает значения с чередующимися знаками, то $E_n(f) \geqslant min|f(x_i) - P(x_i)|$.

Постановка задачи.

С помощью алгоритма Ремеза найти многочлены наилучшего равномерного приближения 5-й и 10-й степени для функции $f(x) = \frac{A}{x^2+px+q}$ на отрезке $[a,b]$. Значения a, b, A, p, q берутся из п/р №4.

Выполнение работы.

По условию:

a = -1, b = 6, A = 1000, p = -6, q = 56.

Тогда, если подставить значения, то $f(x) = \frac{1000}{x^2-6x+56}$.

Была реализована функция f(), которая вычисляет значения в функции f(x). Также была реализована функция Rm(), которая находит точку глобального максимума. Реализована функция R(), которая находит разность значений исходной функции и полинома в конкретной точке. Реализована функция solve_system(), которая находит коэффициенты полинома приближения.

Реализована функция remez(), которая выполняет алгоритм Ремеза.

Разработанный код см. в Приложении А.

Построение полинома приближения 5-ой степени с точностью 0.0000001. (см. табл. 1) σ – уровень квазиальтернанса; R_{max} – глобальный максимум погрешности; ε – точность выравнивания; i_{after} – номер точки квазиальтернанса, за которой идёт точка максимума (от 0).

Таблица 1 – Построение полинома приближения 5-ой степени

Номер шага	Значение σ	Значение R_{max}	Значение ε	Значение i_{after}
1	-0.0053584	-0.011875	0.0065167	6
2	-0.0056254	-0.010897	0.0052719	
3	-0.0058495	0.0085617	0.0027122	5
4	-0.0062292	0.0074734	0.0012442	0
5	-0.0064001	-0.0074765	0.0010764	4
6	-0.0065964	0.0067923	0.00019585	3
7	-0.0066332	-0.0066881	5.4894e-05	1
8	-0.0066422	-0.0066422	2.4078e-14	5

Значения ε и R_{max} по модулю уменьшаются с увеличением числа шагов. Причина такого явления – лучшая точность приближения на большем шаге.

Был построен график исходной функции и полиномов приближения 5-ой степени на 1-ой, 2-ой, 3-ей, 4-ой, 6-ой и последней итерациях. (см. рис. 1)

Кривые накладываются друг на друга в связи с хорошим приближением полиномов.

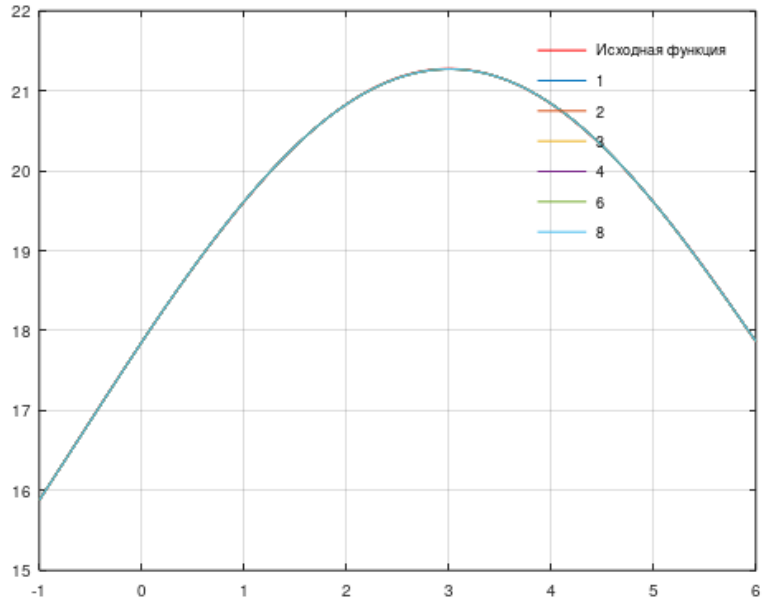

Рисунок 1 – Графики исходной функции и полиномов приближения 5-ой степени

Полином приближения 5-ой степени:

$P = 0.00320064157526569 4x^5 + 0.0023894160814919 83x^4 - 0.0610076147587674 8x^3 - 0.1102013708643497x^2 + 1.928056369773648x + 17.85483881097042$

Построение полинома приближения 10-ой степени с точностью 0.0000001. (см. табл. 2) σ – уровень квазиальтернанса; R_{max} – глобальный максимум погрешности; ε – точность выравнивания; i_{after} – номер точки квазиальтернанса, за которой идёт точка максимума (от 0).

Таблица 2 – Построение полинома приближения 10-ой степени

Номер шага	Значение σ	Значение R_{max}	Значение ε	Значение i_{after}
1	3.3569e-06	8.8936e-06	5.5367e-06	
2	3.4327e-06	-7.3779e-06	3.9452e-06	11
3	3.4876e-06	-6.8612e-06	3.3736e-06	0
4	3.6551e-06	6.0688e-06	2.4137e-06	1

Продолжение таблицы 2

5	3.813e-06	-5.4098e-06	1.5968e-06	2
6	3.9488e-06	5.421e-06	1.4722e-06	10
7	4.0223e-06	5.0208e-06	9.9849e-07	3
8	4.1099e-06	-4.6503e-06	5.4037e-07	4
9	4.1611e-06	-4.6572e-06	4.9611e-07	9
10	4.19e-06	4.3832e-06	1.9315e-07	5
11	4.2082e-06	4.2522e-06	4.4073e-08	8

Значения ε и R_{max} по модулю уменьшаются с увеличением числа шагов, поскольку полином все больше приближается к исходной функции.

Был построен график исходной функции и полиномов приближения 10-ой степени на 1-ой, 2-ой, 3-ей, 5-ой, 7-ой, 10-ой и последней итерациях. (см. рис. 2) Кривые накладываются друг на друга, так как полиномы имеют хорошее приближение.

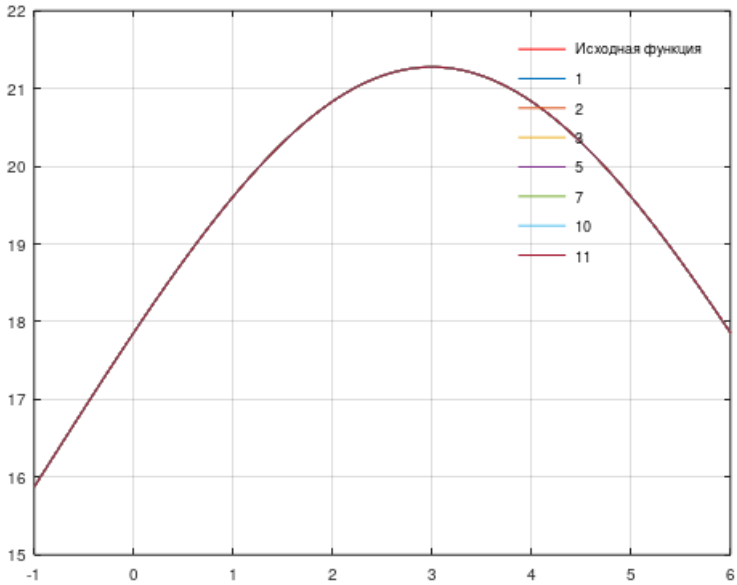

Рисунок 2 – Графики исходной функции и полиномов приближения 10-ой степени

Полином приближения 10-ой степени:

$P = -3.248609149828635e\text{-}08x^{10} + 1.008399245189251e\text{-}06x^{9} - 1.056855200176621e\text{-}05x^{8} + 3.150953725938876e\text{-}05x^{7} +$

$6.79755829740003e\text{-}05x^6 + 0.00050271812418856 14x^5 - 0.00285820000367564 8x^4 - 0.0464005244462626 4x^3 - 0.1139198046899406x^2 + 1.913278926126457x + 17.85714593273138$

Выводы.

Таким образом, были получены навыки применения алгоритма Ремеза. Были получены полиномы приближения 5 – ой и 10 – ой степеней с использованием GNU Octave. Полученные полиномы с помощью алгоритма Ремеза имеют высокую приближенность к исходной функции.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Файл remez.m

```
function remez()
  format long g
  a = -1; b = 6; n = 10;
  k = 1:n+2;
  t = (a+b)/2 + (b-a)/2*cos(pi*(2*k-1)/(2*n+4));
  t = sort(t);
  eps = 0.0000001;
  xp = linspace(a,b,100);
  res = solve_system(t, n+2);
  sygma = res(length(res));
  res = res(1:length(res)-1);
  mp = Rm(xp, res);
  h = mp(1);
  tmax = mp(2);
  i = 1;
  check = 1;
  for i = 1:length(t)
    if (tmax > t(i))
      check++;
    endif
  endfor
  i = check - 1;

  Rmax = R(tmax, res);
  if (i > 0)
  if (sign(Rmax) == sign(R(t(i), res)))
   t(i) = tmax;
   elseif (sign(Rmax) == sign(R(t(i+1), res)))
   t(i+1) = tmax;
  endif
  endif
  if (tmax < t(1) && sign(Rmax) == sign(R(t(1), res)))
    t(1) = tmax;
  elseif (tmax < t(1) && sign(Rmax) != sign(R(t(1), res)))
    tempt = 1:length(t);
    tempt(2:length(t)) = t(1:length(t)-1);
    tempt(1) = tmax;
    t = tempt;
  endif
  if (tmax > t(n+2) && sign(Rmax) == sign(R(t(n+2), res)))
    t(n+2) = tmax;
  elseif (tmax > t(n+2) && sign(Rmax) != sign(R(t(n+2), res)))
    tempt = 1:length(t);
    tempt(1:length(t)-1) = t(2:length(t));
    tempt(n+2) = tmax;
    t = tempt;
  endif
  t = sort(t);

  step = 1;
  plot(xp, f(xp), "r");
```

```
  hold on
  grid on
  plot(xp, polyval(flip(res), xp));
  #printf("%d %.5d %.5d %.5d %d\n", step, sygma, Rmax, abs(h) -
abs(sygma), i);
  while(eps < (abs(h) - abs(sygma)))
    step++;
    res = solve_system(t, n+2);
    sygma = res(length(res));
    res = res(1:length(res)-1);
    mp = Rm(xp, res);
    h = mp(1);
    tmax = mp(2);
    i = 1;
    check = 1;
    for i = 1:length(t)
      if (tmax > t(i))
        check++;
      endif
    endfor
    i = check - 1;

    Rmax = R(tmax, res);
    if (i > 0)
    if (sign(Rmax) == sign(R(t(i), res)))
     t(i) = tmax;
     elseif (sign(Rmax) == sign(R(t(i+1), res)))
     t(i+1) = tmax;
    endif
    endif
    if (tmax < t(1) && sign(Rmax) == sign(R(t(1), res)))
      t(1) = tmax;
    elseif (tmax < t(1) && sign(Rmax) != sign(R(t(1), res)))
      tempt = 1:length(t);
      tempt(2:length(t)) = t(1:length(t)-1);
      tempt(1) = tmax;
      t = tempt;
    endif
    if (tmax > t(n+2) && sign(Rmax) == sign(R(t(n+2), res)))
      t(n+2) = tmax;
    elseif (tmax > t(n+2) && sign(Rmax) != sign(R(t(n+2), res)))
      tempt = 1:length(t);
      tempt(1:length(t)-1) = t(2:length(t));
      tempt(n+2) = tmax;
      t = tempt;
    endif
    t = sort(t);

    #printf("%d %.5d %.5d %.5d %d\n", step, sygma, R(tmax, res),
abs(h) - abs(sygma), i);
    if((n == 5) && (step == 2 || step == 3 || step == 4 || step == 6))
      plot(xp, polyval(flip(res), xp));
    endif;
    if((n == 10) && (step == 2 || step == 3 || step == 5 || step == 7
|| step == 10))
      plot(xp, polyval(flip(res), xp));
    endif
```

```
  endwhile
  plot(xp, polyval(flip(res), xp));
  if (n == 5)
    legend("Исходная функция","1", "2", "3", "4", "6", "8")
  endif
  if (n == 10)
    legend("Исходная функция","1", "2", "3", "5", "7", "10", "11")
  endif
  legend('boxoff')
  hold off
  res;
endfunction

function res = R(xp, k)
  res = f(xp) - polyval(flip(k), xp);
endfunction

function res = solve_system(x, n)
  y = (f(x))';
  A = [];
  for i=1:n
    for j=1:n
      if j==n
      A(i,j) = (-1)^(i-1);
      else
      A(i,j) = x(i)^(j-1);
      endif
    endfor
  endfor
  res = inv(A)*y;
endfunction

function htm = Rm(x, p)
  r = abs(f(x) - polyval(flip(p), x));
  h = max(r);
  j = 0;
  for i = 1:length(r);
    if h == r(i)
      j = i;
      break;
    endif
  endfor
  htm = [h, x(j)];
endfunction

function f = f(x)
  f = (x.^2-6*x+56).**(-1)*1000;
endfunction
```