Отчёт по лабораторной работе
по численным методам
Кобака Ф.А. 18ДКК-1
Вариант 3

Исходный код:

Формулы прямоугольников:

```
% xVec — узлы сетки
% expr — интегрируемая функция в виде символьного выражения
% ftype — тип формулы 1 — центральные 2 — правые 3 - левые
% info — переменная указывает показывать ли отладочную инфу
function res = rectangFun(expr , xVec , fType, info)

    % узнаем чило полученных узлов
    arrSize = numel(xVec);
    % инициализируем переменную в которую будем сумировать
    res = 0;

    % бежим от второго и до последнего узла
    for (i = 2:arrSize)
        % узнаем расстояние между текущим и предыдущим узлом
        h = xVec(i) - xVec(i-1);

        if(fType == 1)
            % для формулы центральных прямоугольников надо взять точку между узлами
            Xi = (xVec(i) + xVec(i-1)) / 2;
        elseif(fType == 2 )
            % для формулы правых берем текущую точку
            Xi = xVec(i);
        else
            % для формулы левых точку левее текущей
            Xi = xVec(i-1);
        end

        % вычисляем значение в выбранной точке
        pY = subs(expr , Xi);
        % считам площадь прямоугольничка
        recSize = h * pY;
        % досумируем к предыдущим площадям
        res = res + recSize;

        %отладочная инфа++++++++++++++++++++++++++++++++++++++++++++
        if(info)
            disp(['+++++++iter ' , num2str(i-1) , '++++++++++']);
            disp(['point in x ' , num2str(Xi)]);
            disp(['point in y ' , num2str(pY)]);
            disp(['size of rectangle ' , num2str(recSize)]);
            disp(['sum ' , num2str(res)]);
            disp(['+++++++iter ' , num2str(i-1) , '++++++++++']);
        end
        %отладочная инфа++++++++++++++++++++++++++++++++++++++++++++

    end

end
```

Трапеции:

```
% xVec — узлы сетки
% yVec — значения в узлах сетки
% info — переменная указывает показывать ли отладочную инфу
function res = trapezFormulaArr(xVec, yVec, info)
    % узнаем чило полученных узлов
    arrSize = numel(xVec);
    %суммируем первый и последний элементы деленные на два (остальные в цикле
    % так как их на два делить не надо)
    summa = (yVec(1) + yVec(arrSize)) / 2;
    %отладочная инфа++++++++++++++++++++++++++++++++++++++
    if(info)
        disp('extreme points data++++++++++++++++++++++');
        disp(['f(x(0)) / 2  = ', num2str(yVec(1)/2) ]);
        disp(['f(x(' , num2str(arrSize) , ')) / 2 = ' , num2str(yVec(arrSize) /
2 )]);
        disp(['summa value is ' , num2str(summa)]);
        disp('extreme points data++++++++++++++++++++++');
    end
    %отладочная инфа++++++++++++++++++++++++++++++++++++++

    %цикл в котором суммируются все другие элементы
    for(i = 2:arrSize-1)
        summa = summa + yVec(i);
        %отладочная инфа++++++++++++++++++++++++++++++++++++++
        if(info)
            disp(['point ' , num2str(i-1) , ' data+++++++++++++++']);
            disp(['f(x(' , num2str(i-1) , ')) = ' , num2str(yVec(i))]);
            disp(['summa value is ' , num2str(summa)]);
            disp(['point ' , num2str(i-1) , ' data+++++++++++++++']);
        end
        %отладочная инфа++++++++++++++++++++++++++++++++++++++
    end
    %тут просто по формуле вычисление результата
    %arrSize – 1 потому, что надо делить на чило промежутков между узлами но не на
число узлов
    res = summa* ((xVec(arrSize) - xVec(1)) / (arrSize - 1));

end
```

Симпсон:

```
% xVec — узлы сетки
% yVec — значения в узлах сетки
% info — переменная указывает показывать ли отладочную инфу
function res = simpsonFormula(x , y, info)
    % узнаем чило полученных узлов
    arrSize = numel(x);
    %суммируем первый и последний элементы деленные на два (остальные в цикле
    % так как их надо на некоторые штуки домножать)
    res = y(1) + y(arrSize);
    %отладочная инфа++++++++++++++++++++++++++++++++++++++
    if(info)
        disp(['y(0) is ' , num2str(y(1))]);
        disp(['y(n) is ' , num2str(y(arrSize))]);
        disp(['y(0) + y(n) is ', num2str(res)]);
    end
    %отладочная инфа++++++++++++++++++++++++++++++++++++++

    for(i = 2:arrSize - 1)
```

```
        if(~mod(i, 2))
            fact = 4; %in case 2,4,6,… в случае четного множим на 4
        else
            fact = 2; % in case 3, 5 ,… в случе нечетного множим на 2
        end

        additElement = fact * y(i);% само непосредсвенно умножение
        res = res + additElement;% прибавляем новый элемент
        %отладочная инфа+++++++++++++++++++++++++++++++++++++++
        if(info)
            disp([num2str(fact),'*y(', num2str(i-1) , ') = ',
num2str(additElement)]);
            disp(['res at step No' , num2str(i-1) , ' is ', num2str(res)]);
        end
        %отладочная инфа+++++++++++++++++++++++++++++++++++++++


    end
    % вычисляем результат по формуле
    res = res * (x(arrSize) - x(1))/(3*(arrSize-1));

end
```

Скрипт результат:

```
% формируем символьное выражение соответсвующее данной функции
f = sym('(3.8 - x^2) / (x^3 +1.5)')

% аналитическое решение сначала найду
% насколько я помню, встроенная int ищет именно аналитически, что подтверждает
необходимость
% преобразовывать в double
analSol = double(int(f , 0 ,1));
disp(['analitic solution ' , num2str(analSol)]);

% дальше нам понадобяться узлы
% 13 узлов между которыми 12 интервалов с дилинной 1/12
x = 0:(1/12):1;

% правые + вывод + отклонение от аналитического решения
right = rectangFun(f , x , 2 , false);
disp(['right rectangles formula result ' , num2str(right) , '. error ' ,
num2str(abs(right – analSol))]);
% левые + вывод + отклонение от аналитического решения
left = rectangFun(f , x , 3 , false);
disp(['left rectangles formula result ' , num2str(left),'. error ' ,
num2str(abs(left – analSol))]);
% центральные + вывод + отклонение от аналитического решения
cent = rectangFun(f , x , 1 , false);
disp(['central rectangles formula result ' , num2str(cent) ,  '. error ' ,
num2str(abs(cent - analSol))]);

% в трапециях и симсоне сделал так, что не используется аналитический вид функции
% но нужна таблица ее значений, найдем ее
y = subs(f , x);

% трапеции + вывод + отклонение от аналитического решения
trap = trapezFormulaArr(x , y , false);
```

```
disp(['trapezoidal formula result ' , num2str(trap) , '. error ' , num2str(abs(trap
– analSol))]);
% симсон + вывод + отклонение от аналитического решения
simpson = simpsonFormula(x , y , false);
disp(['simpson formula result ' , num2str(simpson) , '. error ' ,
num2str(abs(simpson - analSol))]);
```

Результат выполнения:

```
f =

-(x^2 - 3.8)/(x^3 + 1.5) % 5 раз перепроверил, вроде как в условии

analitic solution 2.0517
right rectangles formula result 1.9916. error 0.060131
left rectangles formula result 2.1094. error 0.057647
central rectangles formula result 2.0523. error 0.00062136
trapezoidal formula result 2.0505. error 0.0012419
simpson formula result 2.0517. error 4.5399e-006
```

Выводы:

Лучше всего справилась формула Симсона — самая высокая точность, что не удивительно, ведь она основана на кусочно квадратичной интерполяции, в то время как остальные кусочно-линейные. В плане реализации они все достаточно простые, так что если не учитывать возможные нюансы с вычислительной сложностью метод Симсона - лучший выбор для решения данной задачи. Неплохие результаты у центральных прямоугольников и трапеций, но у центральных лучше, но эти видимо специфика именно этого условия. Левые и правые прямоугольники, ожидаемо проявили себя хуже всех, но у правых ошибка выше, что обусловлено тем, что данная функция на указном отрезке убывает.