Кобак Федор
Отчет по лабораторной работе №1
дополненный и исправленный с подробными пояснениями и разбором допущенных ошибок

функция для построения лагранжа (без исправлений она сразу была нормально сделана):

```
% xVec – массив узлов по которым будет строиться интреполяция
% y – массив значений функций в соответвующих узлах
% g – просто для отладки
function  c = lagIntPoly(xVec , y , g)

    % функция для проверки корректности введенных данных
      % приведена в конце файла
    if(~isIntDataCorrect(xVec , y))
        disp('data not correct');
        return
    end

      % тут создаю счетчик для того чтобы правильно выбирать элементы массива
      % и ту символюную переменную, которая будет свободной для многочлена
    counter = 1;
    syms x;
    % обнуляем c в кторый будем послдовательно доплюсовывать элементы пногочлена
    c = 0;

      % бежим по узлам интерполяции
    for i = xVec

        tempDen = 1; % временный знаментель
        tempNum = 1; % временный чилитель

            % в этом цикле создается конкретный член интерполяционного многочлена
            %опять бежим по узлам интерполяции и чилитель и множим в чилителе
            % x – (точка узла j)
            % а в знаменателе точка (точка узла i) – (точка узла j)
        for j = xVec

            if(i ~= j)
                tempDen = tempDen * (i - j);
                tempNum = tempNum*(x - j);
            end
        end

            % полученный член умнозаем на соответсвующий y и плюсуем к остальным
        c = c + (tempNum/tempDen)  * y(counter);
        counter = counter + 1;
    end

    % chart
      % просто отлаточная часть позволяла мне сравнить график постороенной
интерполяции и данные в начале узлы соединеынные линией
    if g == 1

         resultY =  subs(c , xVec);

         subplot(1 , 2 ,1);
         plot(xVec, y);
         title('input data');
```

```
        subplot(1 , 2 ,2);
        plot(xVec , resultY);
        title('interpolation data');

    end

end
```

Ошибка была допущена при потстроении многочлена Ньютона
Концептуальная ошибка – я пытался в одну функцию засунуть функционал трех, т.е она строила и первый многочлен и второй еще и определяла какой из них оптимальнее строить, это привело к излишней запутонности кода, в итоге я сам не смог его прочитать.
Ошибка в теоритическом подходе **(это основная ошибка)** – дело в том, что у меня в конспекте буквально первым абзацем написано – *для равноудаленных узлов* , т.е. $x_{i+1}-x_{i-1}=const=h$.
Дальнейшие рассуждения строились так – интераоляцию ищем в виде:
$Pn(x)=a_0+a_1(x-x_0)+a_2(x-x_0)(x-x_1)+...+a_n(x-x_0)(x-x_1)...(x-x_{n-1})$
учитывая то что интерполяция должна удовлетворять $Pn(x_i)=y_i$
Подставим $Pn(x_0)=a_0$ (так как остальные члены попросту обнуляться)

Аналогичной логикой и простыми преобразвованиями получается $a_1=\frac{y_1-y_0}{x_1-x_0}=\frac{d^2 y_0}{h}$ (d – обозначение конечной разности треугольничка не нашел)

И $a_0+a_1(x_2-x_0)+a_2(x_2-x_0)(x_2-x_1)=y_2$ превращается в $a_0+a_1 h+a_2 h 2h=y_2$ от куда $a_2=\frac{y_2-2y_1+y_0}{2!h^2}=\frac{d^2 y_0}{2!h^2}$ и через полную индукцию получают след. Формулу $a_n=\frac{d^k y_0}{k!h^k}$

ее несколько преобразованную я использовал в предыдущей работе, но это справедливо только для равноудалённых узлов!! Но я только через некоторае время поиска ошибки заметил, что в моем варианте даны не равноудаленные.
В новом алоголитме я исходил из формулы где x_0-x_i не превращаяется в h*i, да и точка данная для подстановки находится в конце таблицы потому нодо строить второй многочлен Ньютона. Строить его в виде
$Pn(x)=a_0+a_1(x-x_n)+a_2(x-x_n)(x-x_{n-1})+...+a_n(x-x_n)(x-x_{n-1})...(x-x_0)$ (1)

получаем $Pn(x_n)=a_0=y_n$

далее $P_n(x_{n-1})=a_0-a_1(x_{n-1}-x_n)$ от куда $a_1=\frac{y_{n-1}-a_0}{(x_{n-1}-x_n)}$

аналогично для n-2
$a_2=\frac{y_{n-2}-a_0-a_1(x_{n-2}-x_n)}{(x_{n-2}-x_n)(x_{n-2}-x_{n-1})}$
Ну и в общем случае, чтобы найти a_k надо в числителе от y_{n-k} отнять ранее найденный кусок многочлена с подставленным x_{n-k} и делить на произведение скобочек стоящих в (1) рядом с искомым коэффициентом также с подставленным в них x_{n-k}

И наконец сам алгоритм, он реализует только вторую интерполяционную формулу и ничего лшишнего

Вторая интерполяционная формула Ньютона:

```
% на вход подается массив x по которому строиться интреполяция
% массив y по которому строиться интерполяция
% point точка для которой строиться интерполяция
% resutlt – полученный многочлен
function resutlt = newton2Lab(x , y , point)

      % пребираем x и нахоим тот с которого лучше всего строить второй многочлен
Ньютона
      % ближайший к точке справа
      % сохраняем его индекс в n
    for(i = numel(x):-1:2)

        if(point < x(i) & point > x(i-1))
            n = i;
        end

    end


    symX = sym('1');% это те символьные выражения при которых потом будут стоять
      коэффициенты а
    symbX = sym('x');% имя x уже занято потому символьный x в глобальном
поростансве имен (ну или как это в MatLab называется) запишем в symbX


      % проинициализируем результат нулем, как символьное выражение
      % потом в цикле в него доплюсоем все члены
    result = sym('0');
      % от n-го до первого, просчитываем члены многочлена и складываем их в result
    for(i = n:-1:1)
            % по описанной выше логике формируем числитель
        numenator = y(i) - subs(result, x(i));


            % тут формируется знаменатель
            % сналала присваем 1, и домножаем потом в цикле на нужное количесво
            скобочек
            % в случае же если высчитывается a(1) то n == i он и останеться
                  единицей - протсо не попадем в цикл
        denominator = 1;
        for(j = n:-1:i+1)
            denominator = denominator * (x(n) - x(j));
        end
            % в результат дополюсовываем новый член
        result = result + (numenator / denominator) *symX;
            % будем примножать (x – x(i)) чтобы получить те наборы скобок при
            которых стоят коэффициенты а для следующей итерации
        symX = symX * (symbX - x(i))

    end
```

```
end
```

Сценарий выполнения для данного варианта:

```
x = [0.847 1.546 1.834 2.647 2.91];
y = [-1.104 1.042 0.029 -0.344 -0.449]

disp('Lagrange interpolation poly');
lagPoly = lagIntPoly(x , y, 0)
disp('first Newton intrtpolation poly');
nPoly = newton2Lab(x , y, x(1)+x(2))

disp('L(x1 + x2)')
% тут точки с запятой стояли, это глупая ошибка по невнимательности
subs(lagPoly , x(1) + x(2))
disp('N(x1+x2)')
subs(nPoly , x(1) + x(2))

% обозначим узлы жирными точками
plot(x , y , '.r' , 'MarkerSize' , 20);

hold on
vx = x(1):0.01:x(5);
% проведем через них графики
plot(vx , [subs(lagPoly , vx) ;subs(nPoly , vx) ])

legend( 'points','lagrange' , 'newton')
hold off
```

Результат

y =

-1.1040   1.0420   0.0290   -0.3440   -0.4490

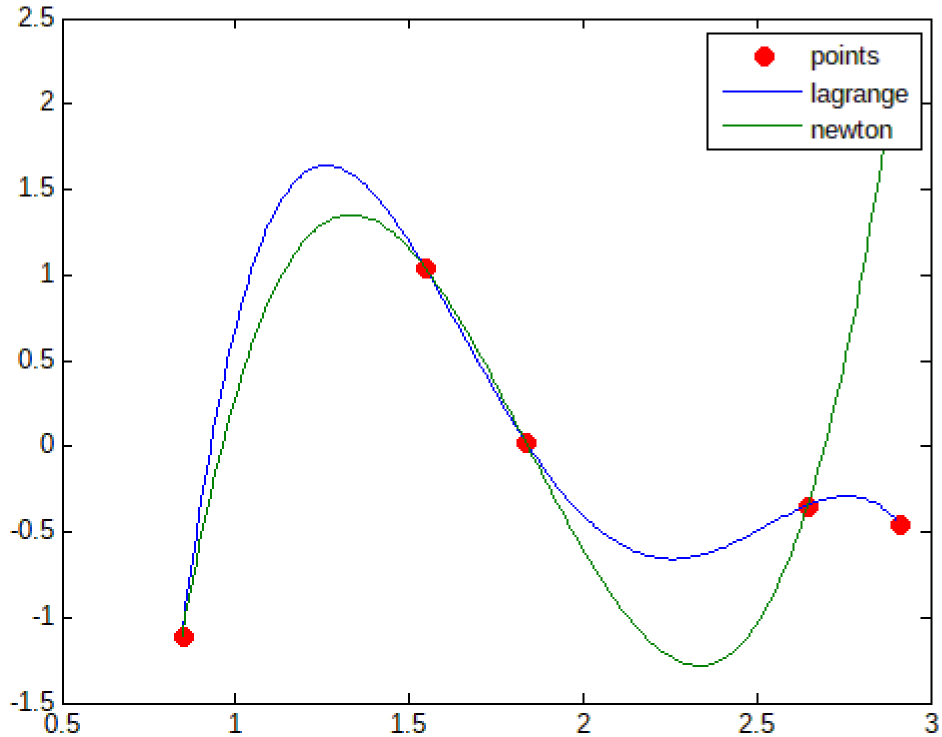
Lagrange interpolation poly

lagPoly =

(774619135907725312*(x - 291/100)*(x - 773/500)*(x - 917/500)*(x - 847/1000))/954192276689337625 - (310748374288564224*(x - 291/100)*(x - 773/500)*(x - 917/500)*(x - 2647/1000))/721117198238440625 + (130604389193744384*(x - 291/100)*(x - 773/500)*(x - 847/1000)*(x - 2647/1000))/1119881916572375375 - (234637540586002 8416*(x - 291/100)*(x - 917/500)*(x - 847/1000)*(x - 2647/1000))/680771132606606375 - (505529058172338176*(x - 773/500)*(x - 917/500)*(x - 847/1000)*(x - 2647/1000))/896564216325879875

first Newton intrtpolation poly

```
ans =

   -0.3440


ans =

   -0.4588


ans =

    2.7780


ans =

    5.2512


nPoly =

(3127738204817169*(x - 917/500)*(x - 2647/1000))/1125899906842624 - (373*x)/813 +
(1478090118844549*(x - 773/500)*(x - 917/500)*(x - 2647/1000))/281474976710656 +
707659/813000

L(x1 + x2)

ans =

   -0.6003

N(x1+x2)

ans =

   -1.2534
```



Лагранж идеально прошел через данные узлы (его я кстате сразу правильно сделал), а Ньютон последнюю точку промахнулся, что не удивительно, ведь мы интерполировали назад от четверной точки

Дополнительные методы

```
function val = isIntDataCorrect( x , y )

    if (numel(x) == numel(y))
        val = true;
    else
        val = false;
    end

end
```