Кобак Федор
Отчет по лабораторной работе №2
Вариант 1

Код основных функций:

Сама функция для построения сплайнов
%использованы также самописные функции
systShow:принимает матрицу, вектор, строку и число ; на основе матрицы и вектора формирует отбражене в командное окно типа СЛАУ, как его пишут на листочке за переменные подставляет строку и нумерует переменные начиная с указанного числа

получается чтото вроде
```
>> systShow([2 3; 4 6] , 'j' , [2 4] , 9)
 /
/ +2*j9+3*j10 = 2
\ +4*j9+6*j10 = 4
 \
```

и функция **swipMetod** – предназначенная для решения СЛАУ методом прогонки ее реализация представлена в конце отчета

```
%x – узлы полученной сетки
%y – значения функции которые соответсвуют узлам сетки
%c_0 – нулевое с (лучше всего производную в нулевом узле)
%c_n – noe с (лучше всего произодную в последнем узле)
% info если true, то выводит дополнителную инфрмацию для отладки

function S = spline3_1forLab(x , y , c_0 , c_n , info)

      %посчитаем количесво элементов в x
    xCount = numel(x);

    %первым сделаем сделаем трехдиагоналку

      %считаем шаг для второго уравнения трехдиаг. Там он используется в левой чати
      %расчет шага для второго уравнения вынес, потому как он не попадал не под
одну из итераций цикла, а править общую логику ооооченьь долго
    h(2) = x(2) - x(1);

    d3MSize = xCount - 2; % это размерность трехдиагональной системы она на 2
меньше размерности x потому что крайние c получены как входные параметры
    d3M = zeros(1 , xCount); % сюда позже запишется трехдиагоналдьная матрица
    bV = [];% а сюда запишеться вектор свободных членов

    for i = 2:xCount-1

        h(i + 1) = x(i + 1) - x(i);% считаем шаг для каждого уравнения (он
используется в левой части)

        if(i ~=2)
            %для первой итерации (i =2) d3M попросту еще не инициализирована
```

```
            d3M = [d3M ; zeros(1 , xCount)]; % в ію строку трехдиагоналки поначалу
запишем нули, а те элементы что не должны быть нулями потом поменяем на не нули
        end

            % тут по формулам заполняем элементы главной диагонали и ближайших
побочных
        d3M(i - 1 , i -1) = h(i);
        d3M(i - 1 , i) = 2 * (h(i) + h(i + 1));
        d3M(i - 1 , i + 1) = h(i+1);

            %по формуле сччитаю соответсвующие элементы правой части
        bV(i-1) = 6*( ((y(i + 1) - y(i))/ h(i+1)) - ((y(i) - y(i -1))/ h(i)));
            %отображение отладочной инфы в случае надобности!!!!!!!!!!!!!!!
        if(info)
            disp(['++++++++++iteration number ' , num2str(i - 1) , '+++++++++++++++
+'])
            disp(['++++++++++ h(', num2str(i - 1) , ') = ' , num2str(h(i))]);
            disp(['++++++++++ h(', num2str(i) , ') = ' , num2str(h(i + 1))]);
            disp(' ');
            disp(['sytem at iteration']);
            systShow(d3M , 'c' , bV, 0 );
            disp(' ');
            disp(['++++++++++iteration number ' , num2str(i - 1) , '+++++++++++++++
+'])
        end
            %отображение отладочной инфы в случае надобности!!!!!!!!!!!!!!!

    end

    %теперь, когда терхдиагональная система составлена самое время заняться ее
решением

      %формируем вектор с и записываем в него уже данные элементы
    c = zeros(1, xCount);
    c(1) = c_0;
    c(xCount) = c_n;

      %в первом и последнем уравнениях системы правим правую часть так как будно мы
перенести c(1) и c(n) с тоящими при них коэффициентами через знак равно
    bV(1) = bV(1) - c(1)*d3M(1);
    bV(d3MSize) = bV(d3MSize) - c(xCount) * d3M(d3MSize, xCount);

      %из 3ех диагоналки удаляем столбцы где до “перенесения” стояти c(0) и c(n)
с коэффициентами
    d3M(: , 1) = [];
    d3M(: , xCount - 1) = [];

      %служебная инфа
    if(info)

        disp(['c0 = ', num2str(c(1))]);
        disp(['c' , num2str(xCount - 1) ,' = ' , num2str(c(xCount)) ]);
        disp(['final 3 diagonal system']);
        systShow(d3M ,'c', bV , 1);

    end

      % методм прогонки решаем ситему уравнений – функция swip metod представлена в
конце документа
```

```
    infoDisp('+++++++++++++++++swip Metod info++++++++++++++++++' , info);
    c(2:xCount -1) = swipMetod(d3M , bV, info);
    infoDisp('+++++++++++++++++swip Metod info++++++++++++++++++' , info);


    if(info)
        disp('c vector is');
        c
    end
      % так как x – уже имеется то в symbolic xSym запишем символьное значение x
    xSym = sym('x'); % its a symbolic expression for good representstion
    for( i = 2:xCount)

            % теперь банально по формуле в цикле делаем все сплайны , на выходе
получаем “вектор” или скорее массив symbolic выражений
        S(i-1) = y(i) + ((1/3) * c(i) * h(i) + 1/6 * c(i-1)*h(i) + (y(i) - y(i-1))/
h(i)) * (xSym - x(i))  + (c(i)/2) * ((xSym - x(i))^2) + ((c(i) - c(i-1))/
(6*h(i)))* ((xSym - x(i))^3);
            % это просто для отладки
        if(info)
            disp(['b(' , num2str(i - 1),')', num2str(((1/3) * c(i) * h(i) + 1/6 *
c(i-1)*h(i) + (y(i) - y(i-1))/h(i)))]);
            disp(['d(' , num2str(i - 1) , ')' , num2str((c(i) - c(i-1))/
(6*h(i)))]);
        end

    end

end
```

Функция для подстановки в сплайны значений
она просто бежит по значениям переданным в val и ищет какой именно сплайн подойдет для каждого этого значения:

```
% x – узлы на которых построен сплайн
% splines – сами сплайны
% val – вектор значений которые требуеотся подставить в сплайны

% result – вектор результатов подстановки в спалайны

function result = splineSubs(x , val , splines)

    xSize = numel(x);
    result = [];
      % пребираем переданные значения для подстановки в сплайн
    for(v= val)
      % эти проверки на тот счучай, если значения за границами сплайна
        if (v <= x(1))
            newVal = subs(splines(1) , v);
        elseif(v > x(xSize))
            newVal = subs(splines(xSize - 1), v);
        else
            % тут перебираем узлы которые использовались для построения сплайна
            for(i = 1:xSize – 1)
                  % и если узел оказался между x и x(i+1) то подставляем и
запоминаем значение
                if(v < x(i+1) && v >= x(i))
```

```
                    newVal = subs(splines(i) , v);
                end
            end
        end
            % сюда сохраняем все значения
        result = [result , newVal];
    end


end
```

Файл сценария описывающий данные именно для моего варианта:

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

% получаем сплайны
disp('its splines!!')
splines = spline3_1forLab(x , y , 0 , 0, false)

% подставляем требуемое число
disp('S(x1+x2)');
splineSubs(x , x(1) + x(2), splines)

 % выведем узлы
plot(x , y , '.r' , 'MarkerSize' , 20);

hold on
%  построим график сплайна
vx = x(1):0.01:x(5);
plot(vx , splineSubs(x , vx , splines) )

legend( 'points', 'splines')
hold off
```

РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ

its splines!!

splines =

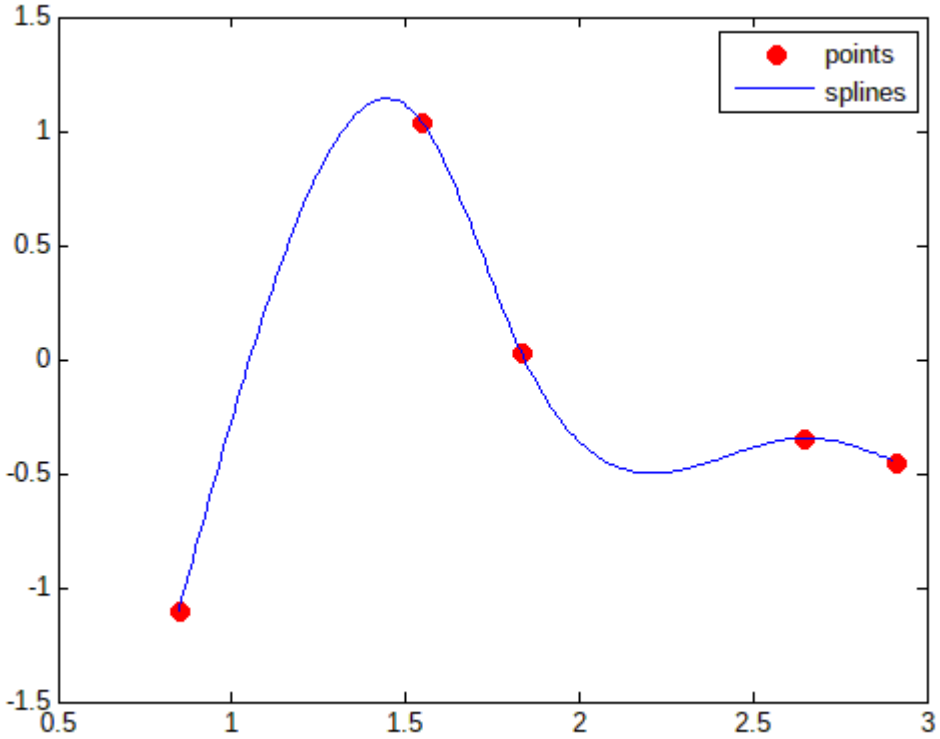
[4715735799299726343/1125899906842624000 - (3083719811811419*(x - 773/500)^2)/281474976710656 - (735269387651745*(x - 773/500)^3)/140737488355328 - (4582856528291995*x)/2251799813685248, (7286755544638545*(x - 917/500)^2)/1125899906842624 - (3745287890496853*x)/1125899906842624 + (709694545424053*(x - 917/500)^3)/35184372088832 + 34507545442348322 49/562949953421312000, (268370747489181*x)/18014398509481984 - (664843541075095*(x - 2647/1000)^2)/281474976710656 - (8155907920409125*(x - 2647/1000)^3)/2251799813685248 - 690733045586566 4603/18014398509481984000, (3370562945881339*(x - 291/100)^3)/1125899906842624 - (682642233492927*x)/1125899906842624 + 740479920646039697/562949953421312000]

S(x1+x2)

ans =

  -0.4408

>>



КОД функций использованный внутри отсновных функций

```
% ТУТ метод прогонки

%a - matrix of system
%b - vecor of free elements

function resVec = swipMetod(a , r , info)

    b(1) = 0;
    c(1) = a(1 , 1);
    d(1) = a(1 , 2);
```

```
    delta(1) = -(a(1 , 2) / (a(1, 1)));
    lambda(1) = r(1)/a(1,1);

    if(info)
        disp(['delta(1) = ' , num2str(delta(1))]);
        disp(['lambda(1) = ' , num2str(lambda(1))]);
    end

    len = numel(r);

    infoDisp('!!!!!!!!!!!!!!straight run!!!!!!!!!!!!!!!!!', info);
    for (i = 2:len - 1)

        b(i) = a(i , i -1);
        c(i) = a(i , i);
        d(i) = a(i , i+1);

        delta(i) = -(d(i) / (c(i) + b(i)*delta(i-1)));
        lambda(i) = (r(i) - b(i) * lambda(i-1)) / (c(i) + b(i)*delta(i-1));

        if(info)
            disp(['===== iteration ' , num2str(i)  , '======'])
            disp(['b = ' , num2str(b(i))]);
            disp(['c = ' , num2str(c(i))]);
            disp(['d = ' , num2str(d(i))]);
            disp(['lambda(i) = ' , num2str(lambda(i))]);
            disp(['delta(i) = ' , num2str(delta(i))]);
            disp(['===== iteration ' , num2str(i)  , '======'])
        end


    end

    b(len) = a(len , len - 1);
    c(len) = a(len , len);

    infoDisp('!!!!!!!!!!!!!!straight run!!!!!!!!!!!!!!!!!', info);

    x(len) = (r(len) - b(len)* lambda(len-1)) / (c(len) + b(len)*delta(len -1));

    infoDisp(['x(' , num2str(len) , ') = ' , num2str(x(len)) ], info);

    infoDisp('!!!!!!!!!!!!!!return run!!!!!!!!!!!!!!!!!' , info);

    for(i = len-1:-1:1)

        x(i) = x(i+1)* delta(i) + lambda(i);
        infoDisp(['x(' , num2str(i) , ') = ' , num2str(x(i)) ], info);

    end

    infoDisp('!!!!!!!!!!!!!!return run!!!!!!!!!!!!!!!!!' , info);

    resVec = x;
end
```

Тут красивая отображалка систем:

```
% function for good presentstion system by matrix
% by Fedor Kobak github.com/Dranikf


% a - matrix of a system
% varName - is a string that will be used for variable name
% b - is a vector of free items
% indexesNumStart - is an start index for system numeration

function systShow(a, varName , b , indexesNumStart)

    sysSize = size(a);
    isEven = ~mod(sysSize(1) , 2);

    disp(' /')
    for (i = 1:sysSize(1))
        equationLine = ' |';

        if(isEven)
            if(((sysSize(1) / 2) == i))
                equationLine = '/ ';
            end
            if((sysSize(1) / 2) == i - 1)
                equationLine = '\ ';
            end
        else
            if (round(sysSize(1) / 2) == i)
                equationLine = '<';
            end
        end

         varInd = indexesNumStart;

        for(j = 1:sysSize(2))
            varL = [varName , num2str(varInd)];

            if(a(i, j) >= 0)
                digit = '+';
            else
                digit = '-';
            end

            equationLine = [equationLine , digit , num2str(abs(a(i , j))) ,'*' ,
varL];
            varInd = varInd + 1;
        end

        equationLine = [equationLine , ' = ' , num2str(b(i))];
        disp(equationLine);
    end
    disp(' \')
end
```