ИНСТИТУТ ТРАНСПОРТА И СВЯЗИ

ФАКУЛЬТЕТ КОМПЬЮТЕРНЫХ НАУК И ЭЛЕКТРОНИКИ

Лабораторная работа

По дисциплине

«Численные методы»

Тема: **Методы приближения функции. Интерполяция и аппроксимация.**

Студент: Виктор Выползов
Группа: 4102BD

Рига
2014 г.

1. Задание

Данная работа разделена на три части: в первой части требуется реализовать аппроксимацию методом МНК, во второй части реализовать индивидуальный метод, а в третьей части работы надо провести сравнение двух выше реализованных методов графическим путем. Для графического вывода можно использовать средства Matlab'a, но так же допустимо применение других средств отображения графиков (к примеру, генерация *.svg файла для просмотра изображения в браузере). Для сравнительного графического анализа требуется на один график нанести одну функцию, которая была обработана двумя различными методами приближения (т.е. получается, что на графике изображены две функции, каждая из которых есть результат работы реализуемых методов приближения).

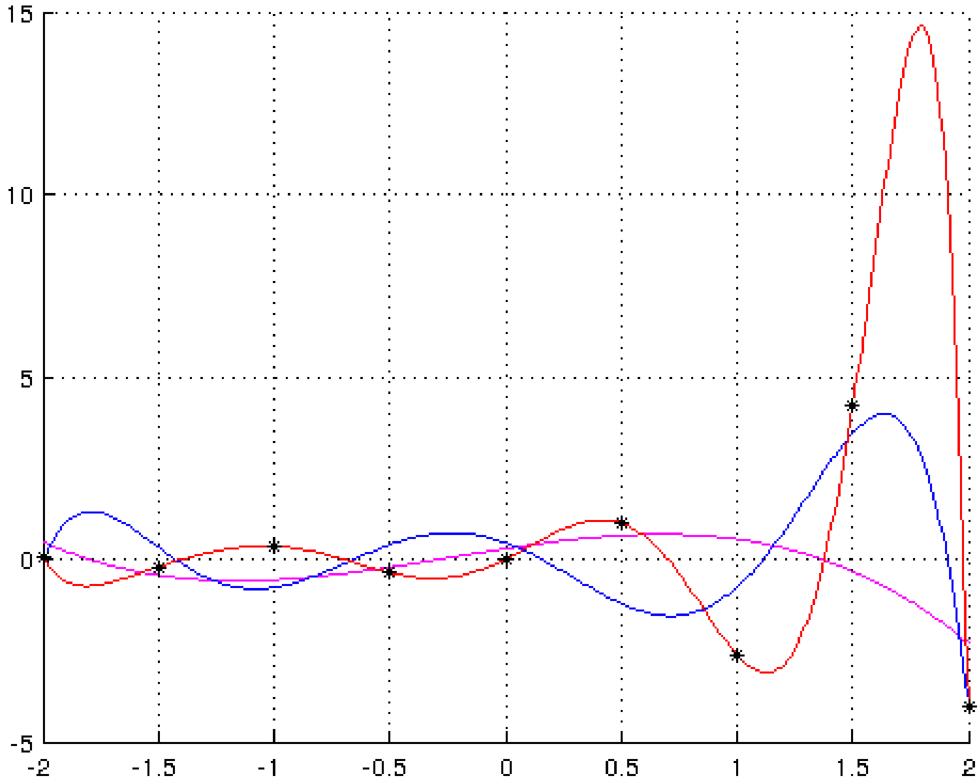
2. Аппроксимация по методу наименьших квадратов

- Листинг программной реализации алгоритма

```
function polynomial = approximation(x, y, k)

    if length(x) == length(y)
        n = k+1;
        m = length(x);
        Max = max(x);
        Min = min(x);

        a = zeros(n,n);
        b = zeros(n,1);

        for i=1:n
           for j=1:n
              if i == 1 && j == 1
                 a(i,j) = m + 1;
              else
                  tmp = 0;

                  for l=1:m
                      tmp = tmp + x(1,l)^((i-1)+(j-1));
                  end

                  a(i,j) = tmp;
              end
           end
        end

        for i=1:n
            tmp = 0;

            for j=1:m
               tmp = tmp + y(1,j)*x(1,j)^(i-1);
            end

            b(i,1) = tmp;
        end

        polynomial = fliplr(gauss(a, b)');

        x1 = Min:1e-2:Max;
        y1 = polyval(polynomial,x1);

        plot(x1,y1,'m');
        grid on;
    else
       disp('Approximation error. Invalid data.');
    end

end
```

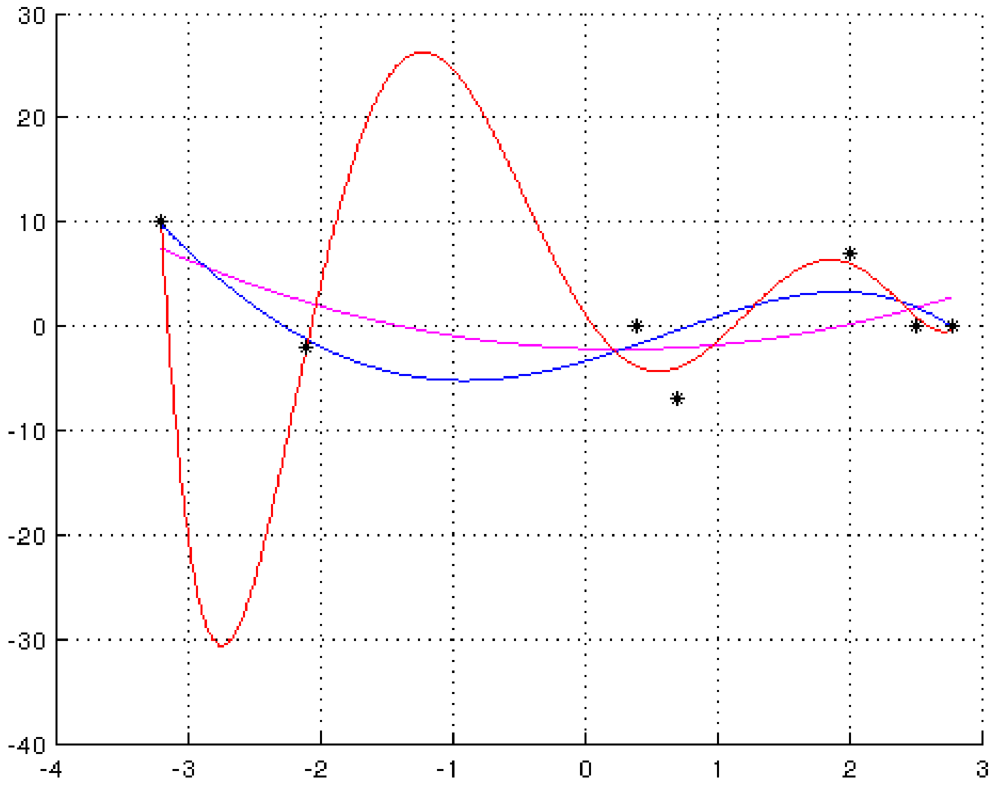
- Графики одной функции при трех различных значениях порядка полинома

$y = \sin(5x)*e^x$; x от -2 до 2, шаг 0.5



Порядок 3 — фиолетовый
Порядок 6 — синий
Порядок 9 — красный

x	-3.2	-2.1	0.4	0.7	2	2.5	2.777	k = 7
y	10	-2	0	-7	7	0	0	



Порядок 2 — фиолетовый
Порядок 4 — синий
Порядок 6 — красный

3. Аппроксимация полиномом Лежандра

- Листинг программной реализации алгоритма

```
function polynomial = legendre_approximation(x, y, k)

    if length(x) == length(y)
        n = k+1;
        m = length(x);
        Max = max(x);
        Min = min(x);

        a = zeros(n,n);
        b = zeros(n,1);

        for i=1:n
            for j=1:n
               tmp = 0;

               for l=1:m
                  tmp = tmp + legendre_polynomial(j-1, x(1,l))*legendre_polynomial(i-1, x(1,l));
               end

               a(i,j) = tmp;
            end
        end

        for i=1:n
           tmp = 0;

           for j=1:m
              tmp = tmp + y(1,j)*legendre_polynomial(i-1,x(1,j));
           end

           b(i,1) = tmp;
        end

        gaussResult = gauss(a,b);

        expression = 0;
        for i=0:n-1
            expression = expression + gaussResult(i+1,1)*legendre_polynomial(i);
        end

        polynomial = pretty_polinomial(expression, n);

        x1 = Min:1e-2:Max;
        y1 = polyval(polynomial,x1);

        plot(x1,y1,'b');
        grid on;

    else
        disp('Legendre approximation error. Invalid data.')
    end

end
```

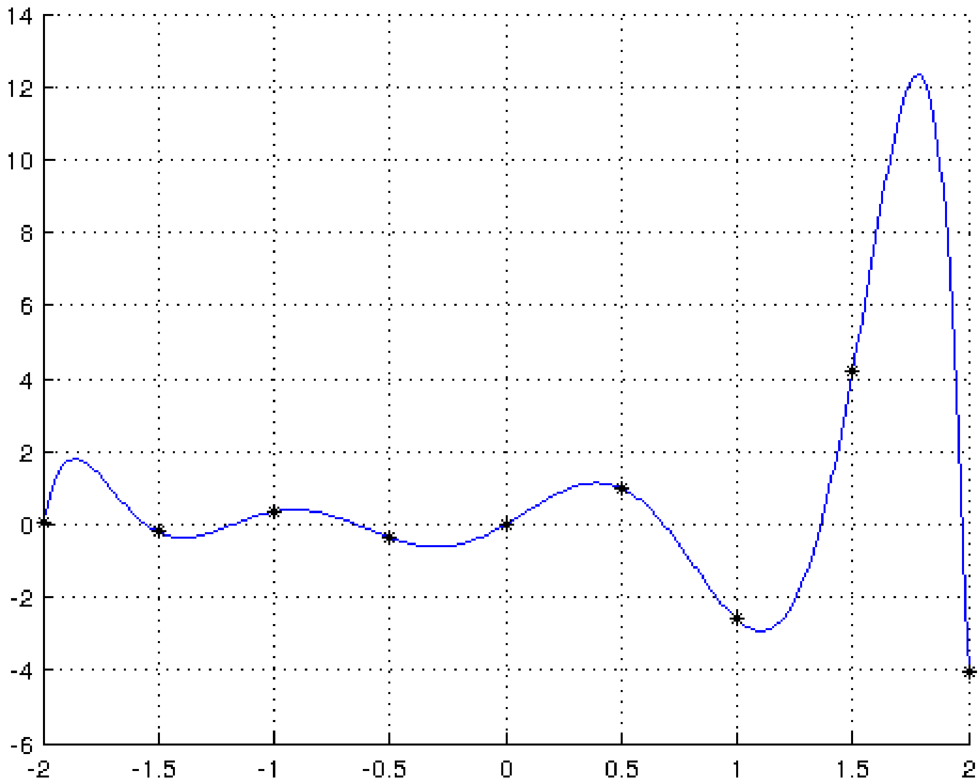
```
function polynomial = legendre_polynomial(n, varargin)

    useExpand = false;

    if isempty(varargin)
        x = sym('x');
        useExpand = true;
    else
        x = varargin{1};
    end

    if n == 0
        polynomial = 1;
    elseif n == 1
        polynomial = x;
    else
        if useExpand
            polynomial = expand(((2*n-1)*x*legendre_polynomial(n-1, x)-(n-
1)*legendre_polynomial(n-2, x))/n);
        else
            polynomial = ((2*n-1)*x*legendre_polynomial(n-1, x)-(n-
1)*legendre_polynomial(n-2, x))/n;
        end
    end

end
```

```matlab
function prettyPolinomial = pretty_polinomial(expression, m)

    expression = char(expression);
    n = length(expression);
    strArray = cell(1,n);
    positionArray = zeros(1,n);
    str = '';
    position = 1;
    isChangedPosition = false;
    i = 1;
    j = 1;

    if expression(1,i) == ' '
       str = [str, expression(i+1)];
       i = i + 2;
    end

    while i <= n
        if expression(1,i) ~= ' '

            if expression(1,i) == 'x'
                isChangedPosition = true;

                if expression(1,i+1) == '^'
                    num = '';
                    l = i + 2;
                    while expression(1,l) ~= ')'
                        num = [num, expression(1,l)];
                        l = l + 1;
                    end
                    i = l;
                    position = str2num(num) + 1;
                else
                    position = 2;
                end
            end

            str = [str, expression(1,i)];
        else
            strArray{1,j} = str;
            positionArray(1,j) = position;

            if isChangedPosition
                position = 1;
                isChangedPosition = false;
            end

            str = '';
            str = [str, expression(1,i+1)];

            i = i + 2;
            j = j + 1;
        end

        if i == n
           strArray{1,j} = str;
           positionArray(1,j) = position;
        end

        i = i + 1;
    end
```

```
    n = m;
    polinomial = zeros(1,n);

    for i=1:n
        m = length(strArray{i});
        str = '';
        j = 1;

        while j<= m
            if strArray{i}(j) ~= '*' && strArray{i}(j) ~= 'x' && strArray{i}(j)
~= '^'
                str = [str, strArray{i}(j)];
            else
               if strArray{i}(j) == '^'
                  j = j + 1;
               end
            end

            j = j + 1;
        end

        if length(str) ~= 0
           polinomial(1, positionArray(1,i)) = str2num(str);
        end
    end

    prettyPolinomial = fliplr(polinomial);

end
```

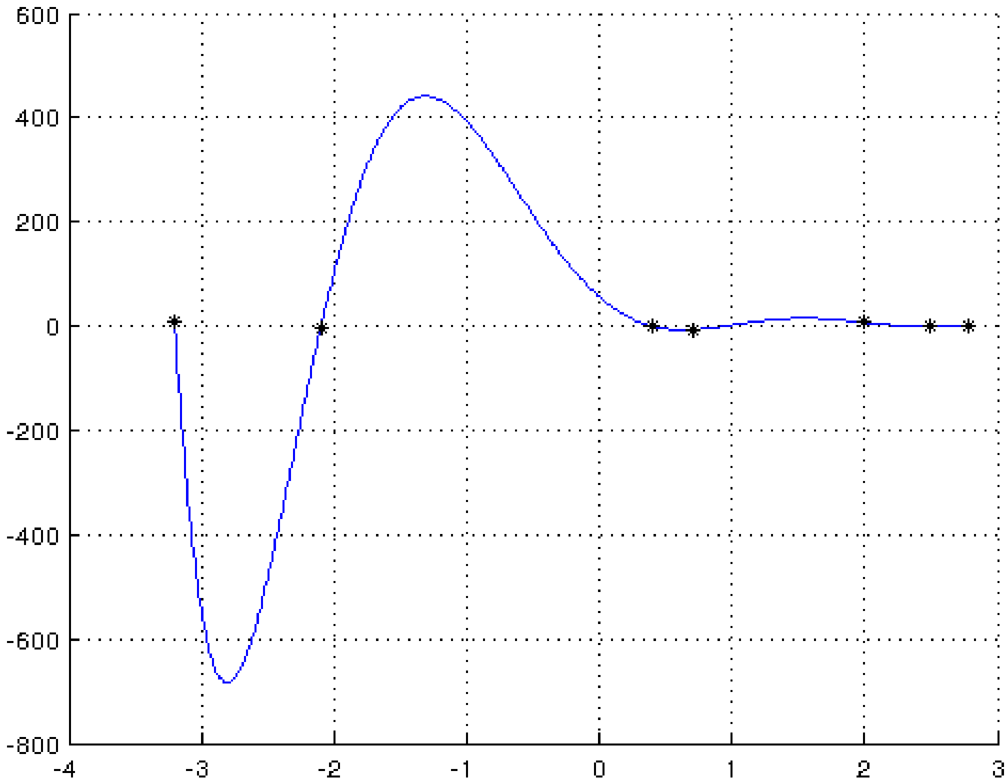
- Требуемая информация для вывода

$y = \sin(5x) * e^x$; x от -2 до 2, шаг 0.5, k = 8



Полином: $0 + 3.7088x + 3.1144x^2 - 11.1407x^3 - 8.7561x^4 + 7.1065x^5 + 5.3652x^6 - 1.1543x^7 - 0.8504x^8$

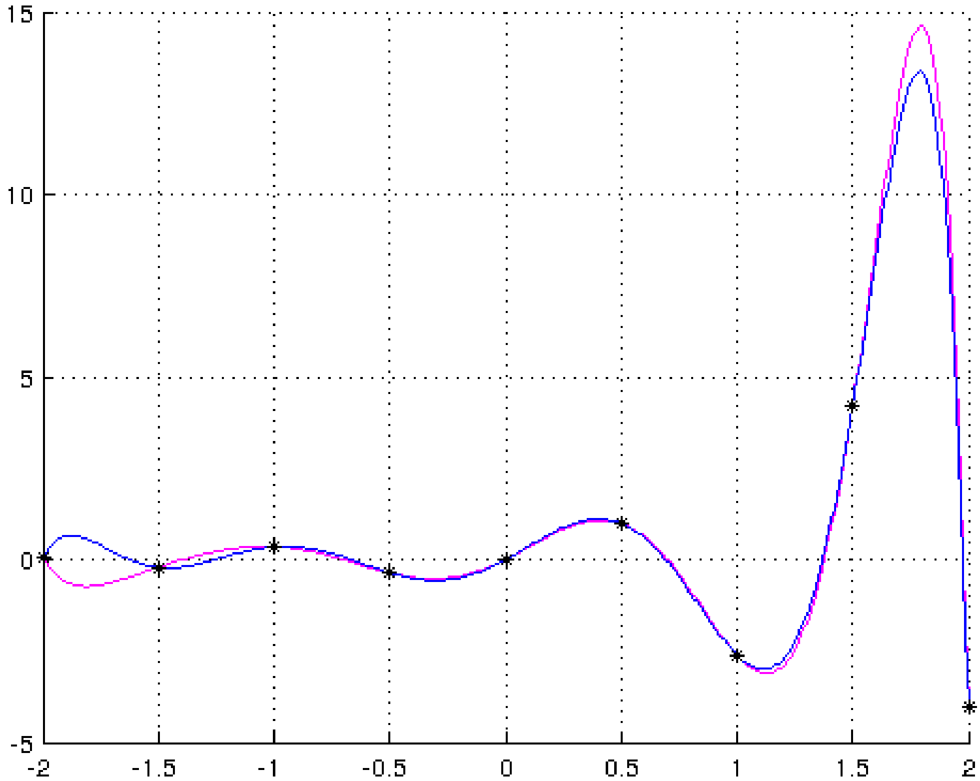
x	-3.2	-2.1	0.4	0.7	2	2.5	2.777	k = 7
y	10	-2	0	-7	7	0	0	



Полином: $56.4086 - 222.076x + 207.6947x^2 + 13.4805x^3 - 71.7475x^4 + 14.3071x^5 + 4.946x^6 - 1.3304x^7$

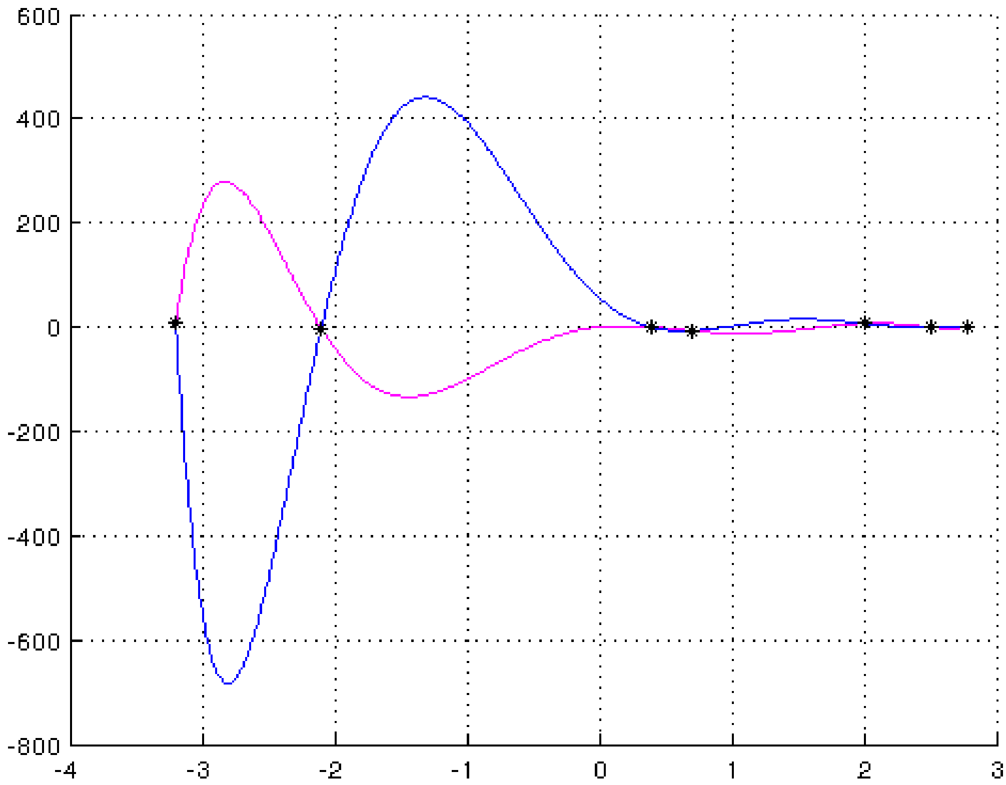
4. Графическое сравнение двух методов

y = sin(5x)*e^x; x от -2 до 2, шаг 0.5, k = 9



Аппроксимация МНК — фиолетовый цвет
Аппроксимация полиномом Лежандра — синий

x	-3.2	-2.1	0.4	0.7	2	2.5	2.777	k = 8
y	10	-2	0	-7	7	0	0	



Аппроксимация МНК — фиолетовый цвет
Аппроксимация полиномом Лежандра — синий

5. Выводы

Как видно из графиков полиномы двух методов не совпадают, лишь некоторые коэффициенты совпадают. Бывают также ситуации когда полиномы совпадают, но это происходит не всегда. Различие алгоритма двух методов не такое большое, и в первом и во втором случае мы строим систему линейных уравнений, находим коэффициенты, получаем полиномы. Но это только на первый взгляд. В методе аппроксимации полиномом Лежандра, необходимо еще:

- реализовать построение полиномов Лежандра
- найти конечный полином, используя полиномы Лежандра, т.е. коэффициентны найденные при решении СЛАУ не являются ответом

Из-за этого метод аппроксимации полиномом Лежандра сложней в реализации, к тому же были проблемы с построением полиномов Лежандра в читаемый вид, но это уже проблема самой реализации программы, а не алгоритма.

Так как и в первом случае и во втором мы получаем аппроксимацию, я советую первый вариант, так как его реализовать программно было намного легче, чего не скажешь об втором методе, хотя алгоритмически понять оба метода не сложно.