# Численные методы

## Основы MATLAB

Юдинцев В. В.

Кафедра теоретической механики

3 октября 2025 г.

САМАРСКИЙ УНИВЕРСИТЕТ
SAMARA UNIVERSITY

# Содержание

1. Интерполирование табличных данных
2. Решение линейных уравнений
3. Задачи линейного програмирования
4. Решение нелинейных уравнений
5. Поиск экстремума функции
6. Численное дифференцирование
7. Численное интегрирование

# Интерполирование табличных данных

# Полиномиальная интерполяция

- Известны значения некоторой функции $y(x)$ в точках $x_0, x_1, x_2, \ldots, x_n$:
$$y_0, y_1, y_2, \ldots, y_n.$$
- Необходимо построить интерполирующий многочлен $L_n(x)$, совпадающий с значениями табличной функции в точках $x_0, x_1, x_2, \ldots, x_n$ и приближающий функцию $y(x)$ на интервале $[x_0, x_n]$.

# Полиномы

Для определения коэффициентов интерполяционного полинома, проходящего через заданные точки с координатами $x_1, f(x_1)$, $x_2, f(x_2)$, ... $x_n, f(x_n)$ используется функция `polyfit`:

```
>> x = [ 1  2  3  4 ];
>> y = [ 0  0.6931  1.0986  1.3863 ];
>> p = polyfit(x, y, 3)
```

Аргументы:

- Список значений $x_0, x_1, x_2, \ldots, x_n$
- Список значений $y_0, y_1, y_2, \ldots, y_n$
- Степень полинома

# Полиномы

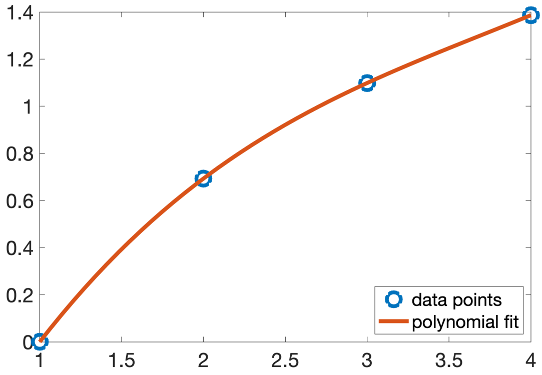
```
>> x = [ 1   2   3   4 ];
>> y = [ 0   0.6931   1.0986   1.3863 ];
>> p = polyfit(x, y, 3)

p =

    0.0283   -0.3137    1.4362   -1.1507

```

Результат – коэффициенты полинома

$$0.0283x^3 - 0.3137x^2 + 1.4362x - 1.1507$$

# Построение графика

```
>> fplot(@(xa) polyval(p,xa),[x(1), x(end)]);
>> print('-dpng', '-r600', 'function.png');
```



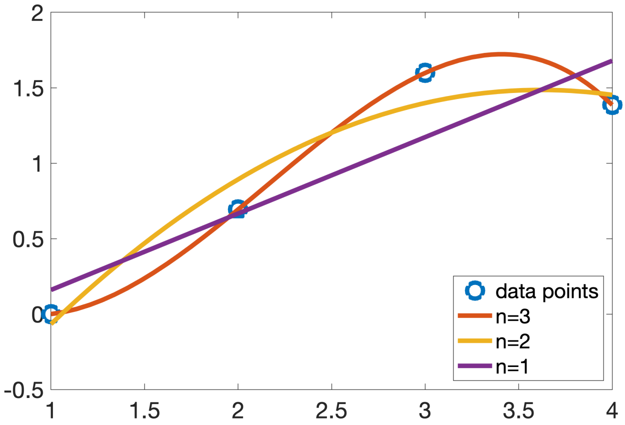
# Полиномы

Если степень полинома, увеличенная на 1, меньше количества точек, то для определения коэффициентов полинома используется метод наименьших квадратов:

```
>> x = [ 1   2   3   4 ];
>> y = [ 0   0.6931   1.0986   1.3863 ];
>> p = polyfit(x, y, 2)

p =

     0.0283    -0.3137     1.4362    -1.1507
```

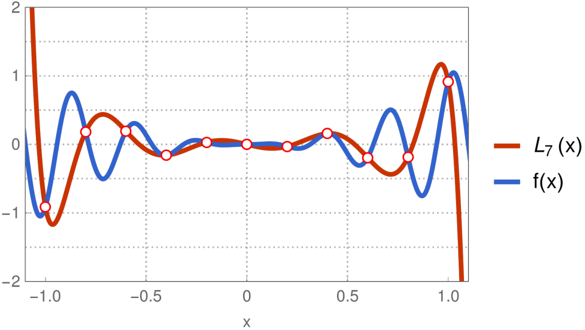
В этом случае полином не будет проходить через узловые точки $(x_i, y_i)$

# Построение графика



# Полиномиальная интерполяция

- Полиномиальная интерполяция используется ”локально”.




- Для глобальной интерполяции используется сплайн-интерполяция – глобальная „кусочно-полиномиальная“ интерполяция.

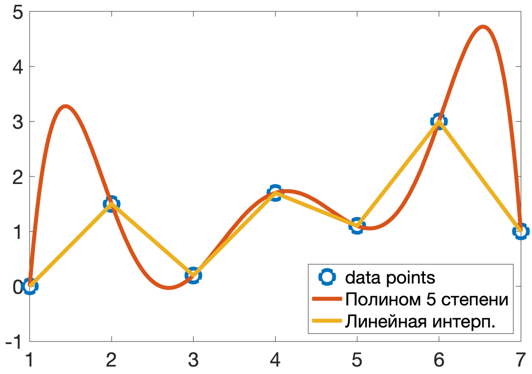
# Сплайн-интерполяция - interp1

```
interp1(x,y,xa,'linear');
```

Функция interp1 вычисляет значение табличной функции $y_i = f(x_i)$, заданной списком значений x и списком значений y, в точке xa, используя линейную интерполяцию.

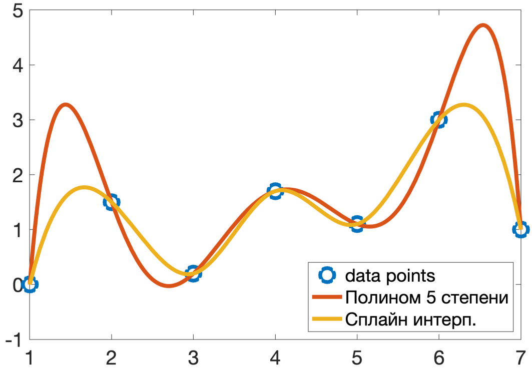
# Сплайн-интерполяция - interp1

```
x = [ 1   2     3     4     5   ];
y = [ 0   0.6   1.0   1.7   1.1];

f = @(xa) interp1(x,y,xa,'linear');
```



# Сплайн-интерполяция

```
x = [ 1   2     3     4     5   ];
y = [ 0   0.6   1.0   1.7   1.1];

f = @(xa) interp1(x,y,xa,'spline');
```



# Решение линейных уравнений

# Системы линейных уравнений

$$\begin{cases} 2x_1 + 3x_2 + x_3 = 5 \\ x_1 + 7x_2 + 2x_3 = 1 \\ 3x_1 + 2x_2 + x_3 = 2 \end{cases}$$

```
A = [2 3 1; 1 7 2; 3 2 1];
B = [5;1;2];

x = A\B

x =
      6.0000
      9.0000
    -34.0000
```

# Системы линейных уравнений

Решение двух систем, отличающихся столбцами правых частей, ”в одно действие”:

$$\begin{aligned} 2x_1 + 3x_2 + x_3 &= 5 \quad & 2x_1 + 3x_2 + x_3 &= 6 \\ x_1 + 7x_2 + 2x_3 &= 1 \quad & x_1 + 7x_2 + 2x_3 &= 7 \\ 3x_1 + 2x_2 + x_3 &= 2 \quad & 3x_1 + 2x_2 + x_3 &= 1 \end{aligned}$$

```
A = [2 3 1; 1 7 2; 3 2 1];
B = [5 6; 1 7; 2 1];

x = A\B

x =
    6.0000     5.0000
    9.0000    10.0000
  -34.0000   -34.0000
```

# Задачи линейного програмирования

# Линейное программирование

Линейное программирование — это математический численный метод для оптимизации моделей, в которых целевые функции

$$f(x_1, x_2, \dots, x_n) \rightarrow min$$

и ограничения, например

$$g(x_1, x_2, \dots, x_n) > 0$$

строго являются линейными уравнениями.

# Пример

| Вид сырья | Прод. 1 | Прод. 2 | Прод. 3 | Прод. 4 | Запас сырья |
|---|---|---|---|---|---|
| Сырье 1 | 4 | 2 | 1 | 8 | $\leq 1200$ кг |
| Сырье 2 | 2 | 10 | 6 | 0 | $\leq 600$ кг |
| Сырье 3 | 3 | 0 | 6 | 1 | $\leq 1500$ кг |
| Прибыль | 15 р. | 6 р. | 12 р. | 24 р. | max |

Прибыль (целевая функция) $S = x_1 s_1 + x_2 s_2 + x_3 s_3 + x_4 s_4 \to max$
Ограничения

$$x_1 a_{11} + x_2 a_{12} + x_3 a_{13} + x_4 a_{14} \leq 1200$$

$$x_1 a_{21} + x_2 a_{22} + x_3 a_{23} + x_4 a_{24} \leq 600$$

$$x_1 a_{31} + x_2 a_{32} + x_3 a_{33} + x_4 a_{34} \leq 1500$$

# Функция linprog

```
x = linprog(f,Ane,Bne,Ae,Be,lb,ub)
```

$$\min_x \boldsymbol{f}^T x$$

При условиях равенствах

$$\boldsymbol{A}_e \mathbf{x} = \boldsymbol{B}_{\mathbf{e}}$$

При условиях неравенствах

$$\boldsymbol{A}_{ne} \mathbf{x} \leq \boldsymbol{B}_{\mathbf{ne}}$$

При условиии

$$\boldsymbol{L} \leq \mathbf{x} \leq \boldsymbol{U}$$

# Пример

| Вид сырья | Прод. 1 | Прод. 2 | Прод. 3 | Прод. 4 | Запас сырья |
|---|---|---|---|---|---|
| Сырье 1 | 4 | 2 | 1 | 8 | $\leq 1200$ кг |
| Сырье 2 | 2 | 10 | 6 | 0 | $\leq 600$ кг |
| Сырье 3 | 3 | 0 | 6 | 1 | $\leq 1500$ кг |
| Прибыль | 15 р. | 6 р. | 12 р. | 24 р. | max |

```
A = [4 2 1 8; 2 10 6 0; 3 0 6 1];
B = [1200; 600; 1500];
f = -[15 6 12 24];
x = linprog(f,A,B,[],[],[0 0 0 0],[])
```

x = [0 0 100 137.5]

# Решение нелинейных уравнений

# Функции одной переменной

Для решения уравнений вида

$$f(x) = 0$$

используется функция `fzero`, первым аргументом которой является ссылка на функцию $f(x)$, вторым – начальное приближение для искомого значения решения уравнения или интервал внутри которого находится решение:

```
>> fzero( @(x) cos(x) - x, 1)

ans =

    0.7391
```

# Система нелинейных уравнений

Для решения системы нелинейных уравнений используется функция `fsolve`. Например, для решения системы

$$\begin{cases} x^2 + y^2 + z^2 - 1 = 0 \\ 2x^2 + y^2 - 4z = 0 \\ 3x^2 - 4y + z^2 = 0 \end{cases}$$

необходимо написать m-файл с векторной функцией векторного аргумента

```
function y = f(x)
    y = [x(1)^2+x(2)^2+x(3)^2-1;
         2*x(1)^2+x(2)^2-4*x(3);
         3*x(1)^2-4*x(2)+x(3)^2];
```

# Система нелинейных уравнений

Для поиска решения необходимо вызвать функцию `fsolve`, передав ей ссылку на функцию и вектор начального приближения $[x_0, y_0, z_0]$:

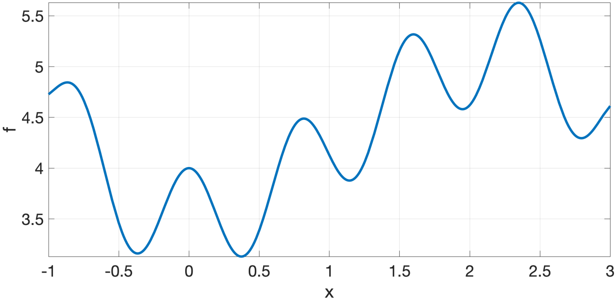
```
>>  fsolve(@y,[1;1;1])

ans =

     0.7852
     0.4966
     0.3699
```

# Поиск экстремума функции

# Функция fminbnd(fun,x1,x2)

Поиск минимума фунции $f(x)$ на интервале $[x_1, x_2]$

```
fun = @(x) x^2 - 0.3*x^3 + 3 + cos(4*x)^2;
fplot(fun,[-1 3]);
fminbnd(fun,-1,3)
ans =
    0.372
```



# Функция нескольких переменных

Найти минимум функции нескольких переменных

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}), \qquad \boldsymbol{x} = [x_1, x_2, \dots, x_n]$$

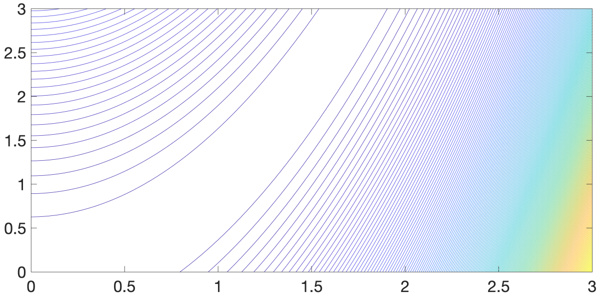
Пример:

$$f(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$

```
fun = @(x1,x2) 100*(x2-x1.^2).^2+(1-x1).^2;
```

# Функция нескольких переменных

```
x1 = linspace(0,3,100);
x2 = linspace(0,3,100);
[X1,X2] = meshgrid(x1,x2);
f = fun(X1,X2);
contour(X1,X2,f,200);
```



# Функция fminsearch

Функция fminsearch

```
fminsearch(@(x) fun(x(1),x(2)),[0, 0.5])

ans =

     1.0000     1.0000
```

Аргументы функции fminsearch

- ссылка на функцию от векторного аргумента
- начальное приближение

# Численное дифференцирование

# Функция diff

Разность первого порядка

$$\Delta y_i = y_{i+1} - y_i$$

```
x1 = 1; xn = 2; h=0.1;
x = x1:h:xn;
y = log(x);
dy = diff(y)

 0          0.0953     0.1823     0.2624     0.3365   ...
 0.0953     0.0870     0.0800     0.0741     0.0690   ...
```

# Функция diff

Разность второго порядка

$$\Delta^2 y_i = \Delta y_{i+1} - \Delta y_i$$

```
d2y = diff(y,2)

-0.0083    -0.0070    -0.0059    -0.0051    ...
```

# Производная первого порядка

Для внутренних узлов

$$y'(x_i) = \frac{\Delta y_{i-1} - \Delta y_i}{2h} = \frac{y_{i+1} - y_{i-1}}{2h}$$

```
dy   = diff(y);

dfi  = @(i) (dy(i-1)+dy(i))/(2*h);

[x(2) dfi(2)]

ans =

      1.3000      0.7708
```

Точное значение 0.7692 (0.2%)

# Производная первого порядка

Для внешнего узла $i = 1$

$$y'(x_1) = \frac{\Delta y_1 - \Delta^2 y_1/2}{2h}$$

Для внешнего узла $i = n$

$$y'(x_n) = \frac{\Delta y_{n-1} - \Delta^2 y_{n-2}/2}{2h}$$

# Вторая производная

$$y''(x_i) = \frac{\Delta y_i - \Delta y_{i-1}}{h^2}$$

```
x     = x1:dx:x2;
y     = some_math_function(x);
dy    = diff(y);
dydt = @(i) (dy(i) - dy(i-1))/dx^2;
```

# Пример

```
dx  =  0.05;
x   =  (0:dx:0.5)';
y   =  sin(x); yp = cos(x); ypp =-sin(x);

dy   =  diff(y);
d2y  =  diff(y,2);

ypa   = @(i) (dy(i-1) + dy(i))/(2*dx);
yppa  = @(i) (dy(i) - dy(i-1))/dx^2;

ind   =  2:numel(x)-1;
res   =  [x(ind) yp(ind) ypa(ind) ypp(ind) yppa(ind)]

    0.1000    0.9950    0.9946   -0.0998   -0.0998
    0.2000    0.9801    0.9797   -0.1987   -0.1986
    0.3000    0.9553    0.9549   -0.2955   -0.2955
    0.4000    0.9211    0.9207   -0.3894   -0.3893
```

# Численное интегрирование

# Формула трапеций

Для функции, заданной таблично, интеграл можно вычислить используя формулу трапеций:

```
>> x = 1.0:0.1:2.0;
>> y = log(x);

>> trapz(x, y)

ans =

     0.3859
```

Точное значение интеграла равно $\log 4 - 1 \approx 0.386294361119891$.

# Функция quad (intergral)

Для известной подинтегральной функции более точный результат дает использование функции quad, в которую нужно передать ссылку на подинтегральную функцию и пределы интегрирования

$$\int_1^2 \ln(x)\,dx$$

```
>> format long
>> quad(@log, 1, 2)

ans =

   0.386294334336416
```

# Функци quad (intergral)

Третим аргументом функции quad является желаемая точность результата (по умолчанию – $10^{-6}$)

```
f = @(x) sin(x).*log(x).^2;

quad(f, 1, 2, 1e-8)

ans =
    0.1822
```

Функция integral

```
integral(f, 1, 2, 'AbsTol', 1e-8)

ans =
    0.1822
```