



*Российский государственный университет  
нефти и газа им. И.М. Губкина*

*Кафедра Информатики*

*Дисциплина: Программные комплексы  
общего назначения*

*Преподаватель:*

**К.Т.Н., ДОЦЕНТ**

**Коротаев**

**Александр Фёдорович**



# Работа с файлами

## Функции **save** и **load**

Функция **save** сохраняет значения заданных переменных в **mat**-файл в текущем каталоге

**save <имя файла> <имена переменных>**

Функция **load** позволяет загрузить из указанного **mat**-файла ранее сохраненные переменные

**load <имя файла> <имена переменных>**

```
S =  
1 2  
>> save dat S  
>> clear  
>> load dat S  
>> S  
S =  
1 2
```



# Работа с файлами

## Функция **load** и текстовый файл

Функция **load** позволяет загрузить данные из подготовленного ранее текстового файла

### Пример

Пусть в файле **d.txt** записаны следующие данные (*числа могут разделяться пробелами или знаками табуляции*):

```
1 2    1.5e-5
3 4    0.33
```

```
>> w=load('d.txt')
w =
    1.0000    2.0000    0.0000
    3.0000    4.0000    0.3300
>> format shortg
>> w
w =
     1     2    1.5e-05
     3     4     0.33
```

# Бинарные и текстовые файлы



**Бинарные** файлы хранят данные в виде двоичных кодов (последовательности байтов). Сам файл не содержит информации о типе данных.

В **текстовых** файлах явным образом записаны числовые данные или текст. Содержимое текстовых файлов интерпретируется как набор символьных строк, разделенных управляющими символами «возврат каретки» и «перевод строки».

Основные этапы работы с файлами:

- **открытие файла**
- **операции ввода/вывода**
- **заккрытие файла**

# Открытие файла



Независимо от типа файла он открывается функцией **fopen** и закрывается функцией **fclose**

**fid=fopen('имя файла ', 'флаг' )**

**флаг** – указывает, для чего открывается файл

'r'	открытие файла для чтения
'r+'	открытие файла для чтения и записи
'w'	удаление содержимого существующего файла или создание нового и открытие его для записи
'a'	добавление записи в конец файла или создание и открытие нового файла для записи



Если к атрибуту **'флаг'** добавляется буква **'t'**, то файл открывается как текстовый. Если ничего не добавляется (или **'b'**), то файл открывается как бинарный

**Возвращаемый идентификатор fid**

используется в качестве числового описателя открытого файла. При неудачном открытии возвращается **-1**

**Примеры**

**fid1=fopen(name,'r+t')** – текстовый файл открывается для чтения и записи

**fid2=fopen(name,'w')** – бинарный для записи

Если файл больше не нужен, его надо закрыть функцией **fclose(fid)**

# Запись и чтение бинарных файлов



**count = fwrite(fid, A, PRECISION, skip)**

где **fid** - числовой идентификатор открытого файла,

**A** – массив записываемых данных,

**PRECISION** – задаёт тип данных (размер памяти)

(по умолчанию, тип 'uint8' – целое без знака, 1 байт),

**skip** - количество байтов, разделяющих элементы  
(необязательный параметр),

**count** – количество записанных в файл элементов  
(возвращаемый параметр)

**[A,count]=fread(fid, size, PRECISION, skip)**

**size** – количество подлежащих прочтению элементов  
(варианты для **size**: 16, [2 3], inf, [1,inf],...)



# Параметр PRECISION

<b>PRECISION</b>	<b>Тип данных</b>	<b>Размер, байт</b>
<b>'char'</b>	<b>СИМВОЛЬНОЕ</b>	<b>1</b>
<b>'short'</b>	<b>целое</b>	<b>2</b>
<b>'int'</b>	<b>целое</b>	<b>4</b>
<b>'long'</b>	<b>целое</b>	<b>8</b>
<b>'ushort'</b>	<b>целое без знака</b>	<b>2</b>
<b>'uint'</b>	<b>целое без знака</b>	<b>4</b>
<b>'ulong'</b>	<b>целое без знака</b>	<b>8</b>
<b>'float'</b>	<b>вещественное</b>	<b>4</b>
<b>'double'</b>	<b>вещественное</b>	<b>8</b>





## Пример

```
a=[1 2 3]; b=[4 5 6; 7 8 9];  
fid1=fopen('data.txt','w');  
fwrite(fid1,a,'double');  
fwrite(fid1,b,'double');  
fclose(fid1);  
fid1=fopen('data.txt','r');  
[A,c1]=fread(fid1,[1,3],'double')  
B=fread(fid1,[2,3],'double')  
fclose(fid1);
```

A =

1 2 3

c1 =

3

B =

4 5 6

7 8 9

# Указатель файла



При открытии файла формируется логический объект – **указатель файла**. Положение указателя определяется как величина смещения в байтах от начала файла. После открытия файла смещение равно 0. После прочтения очередного элемента указатель файла передвигается на соответствующее количество байтов.

Значение смещения возвращает функция **ftell(fid)**

Перед повторным чтением данных из того же файла можно вернуть указатель в начало файла функцией **frewind(fid)**

Для перемещения файлового указателя относительно текущего положения используется функция

**status = fseek(fid, n, ORIGIN)**

где **n** - смещение в байтах вперёд (**n > 0**) или назад (**n < 0**).

**ORIGIN** =  $\left\{ \begin{array}{l} \text{'bof'} \text{ или } -1 \text{ от начала файла} \\ \text{'cof'} \text{ или } 0 \text{ от текущего положения} \\ \text{'eof'} \text{ или } 1 \text{ от конца файла} \end{array} \right.$

**status** : **0** в случае успеха, **-1** в случае неудачи



**Пример** Читаем только **b**, пропустив **a**

```
a=[1 2 3]; b=[4 5 6; 7 8 9];  
fid1=fopen('data.txt','w');  
fwrite(fid1,a,'double');  
fwrite(fid1,b,'double');  
fclose(fid1);  
fid1=fopen('data.txt','r');  
fseek(fid1, 24, 'bof');  
B=fread(fid1,[2,3],'double')  
fclose(fid1);
```

B =

```
4    5    6  
7    8    9
```

# Чтение и запись файлов Excel



Функция **xlsread** считывает данные из **xls**-файла

**A = xlsread(filename, sheet, range)**

где **filename** - имя файла электронной таблицы,

**sheet** - наименование листа в файле,

**range** - диапазон ячеек с данными,

**A** - массив, в который будут считаны данные

Функция **xlswrite(filename, A, sheet, range)** записывает

массив **A** в диапазон ячеек **range** листа **sheet**

**xls**-файла с именем **filename**

**status = xlswrite(filename, A, sheet, range),**

Выходной параметр **status** : **1** в случае успеха,

**0** в случае неудачи

В обеих функциях параметры **sheet** и **range** являются

необязательными. По умолчанию, принимается **Лист1**,

начиная с ячейки **A1**

# Чтение и запись файлов Excel



## Пример

Набираем в окне **Editor** следующую программу:

```
values = {1, 2, 3 ; 4, 5, 'x' ; 7, 8, 9};  
headers = {'First', 'Second', 'Third'};  
xlswrite('myExample.xls', [headers; values]);  
A = xlsread('myExample.xls', 'B3:C4')
```

В файле **myExample.xls** будет следующее содержимое:

First	Second	Third
1	2	3
4	5	x
7	8	9

В командное окно будет выведен результат:

```
A =  
5 NaN  
8 9
```

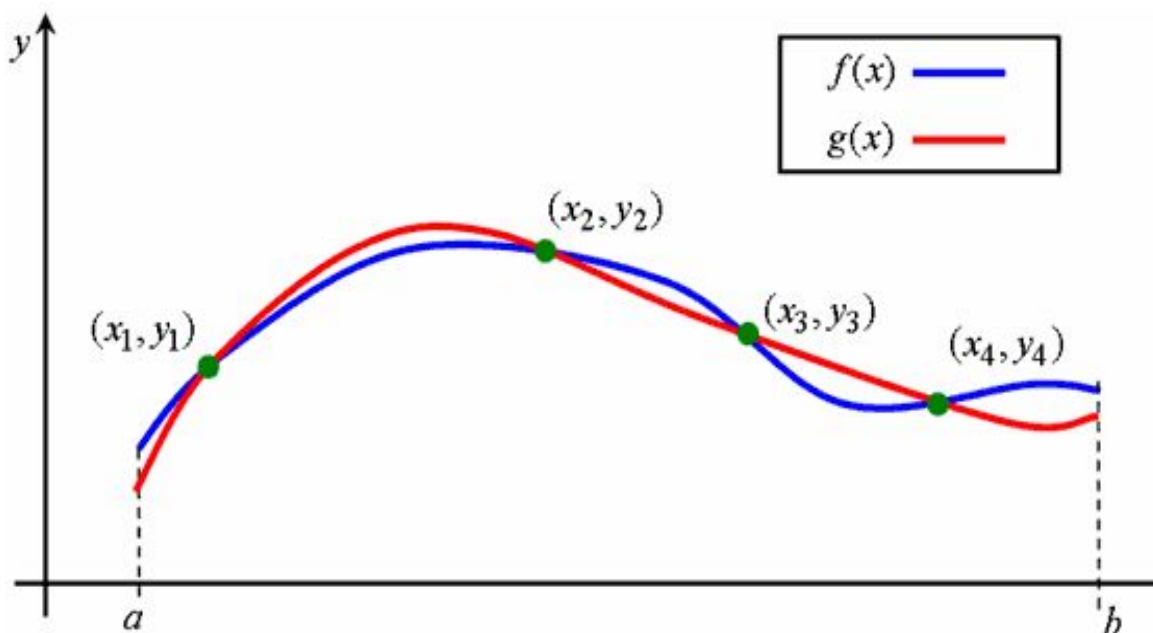
# Интерполяция и аппроксимация данных



Задача **интерполяции** состоит в построении по заданной функции  $f(x)$  другой (более простой) функции  $g(x)$ , совпадающей с заданной в некотором наборе точек  $\{x_1, x_2, \dots, x_{n+1}\}$  из отрезка  $[a, b]$  (эти точки называются **узлами интерполяции**), т.е. должны выполняться условия:

$$g(x_k) = y_k, \quad k = 1, 2, \dots, n+1,$$

где  $y_k$  - известные значения функции  $f(x)$  в точках  $x_k$





# Интерполяция и аппроксимация данных

На практике часто возникает задача о восстановлении непрерывной функции по ее табличным значениям, например, полученным в ходе некоторого эксперимента.

При **линейной интерполяции** узловые точки соединяются друг с другом отрезками прямых. Для повышения точности применяют параболы (**квадратичная интерполяция**) или полиномы более высокой степени (**полиномиальная интерполяция**).

Если применяется **аппроксимация**, то её результат может и не проходить через узловые точки. При этом, степень приближения оценивают по **методу наименьших квадратов**, т.е. минимизируя сумму квадратов отклонений значений искомой функции от заданных в узловых точках.



# Слайн - интерполяция

## Интерполяция кубическими сплайнами:

$y_i = \text{spline}(x, y, x_i)$ , где  $x$  – вектор узлов интерполяции;  $y$  – вектор значений в узлах;

$x_i$  – вектор аргументов, при которых вычисляются значения искомой функции

Промежуточные точки ищутся по отрезкам полиномов третьей степени — это кубическая сплайновая интерполяция. Такие полиномы вычисляются так, чтобы не только их значения совпадали с координатами узловых точек, но и чтобы в узловых точках были непрерывны производные 1 и 2 порядков.



# Интерполяция кубическими сплайнами



## Пример 1

```
>>x=[1,2,3,4,5,6];  
>> y=[6.5,20,53.5,167,473,1470];  
>>xi =[1.5,2.5,3.5,4.5,5.5];  
>> yi=spline(x,y,xi)  
yi =  
15.233 29.766 98.700 270.995 847.754
```

## Пример 2

```
>> x=[1,2,3,4,5];  
>> xi=[1.5,1.8,2.3,3,3.5];  
>> yi=spline(x,sin(x),xi)  
yi =  
1.0222 0.9843 0.7358 0.1411 -0.3414
```



## 1D-интерполяция (поиск в таблице)

$y_i = \text{interp1}(x, y, x_i, \text{метод})$ , где  $x, y$  – векторы значений узлов и функции,  $x_i$  – вектор значений аргументов, задаваемый пользователем, **метод** – аргумент, позволяющий выбрать метод интерполяции.

Методы интерполяции:

**'nearest'**- ступенчатая (метод ближайшего соседа)

**'linear'**- кусочно-линейная

**'spline'**- кубическими сплайнами

**'pchip'**- кусочно-кубическими полиномами Эрмита

# Пример



x	2.5	3.7	8.4	11.7	20	27	38
y	1.4	2.7	5.6	7.5	9.1	13.2	15.3

Вычислить при  $x_i = 3, 4, 6, 10, 25, 30$

```
>> yi1=interp1(x,y,xi, 'nearest')
```

```
yi1 =
```

```
1.4000  2.7000  2.7000  5.6000  13.2000  13.2000
```

```
>> yi2=interp1(x,y,xi, 'linear')
```

```
yi2 =
```

```
1.9417  2.8851  4.1191  6.5212  12.0286  13.7727
```

```
>> yi3=interp1(x,y,xi, 'pchip')
```

```
yi3 =
```

```
1.9884  2.9419  4.2606  6.6349  12.2020  14.0582
```

```
>> yi4=interp1(x,y,xi, 'spline')
```

```
yi4 =
```

```
1.9912  2.9666  4.3323  6.5662  11.8137  15.1007
```

# Полиномиальная регрессия



Функция **аппроксимации** данных полиномами по методу наименьших квадратов

**polyfit(x,y,n)** - возвращает коэффициенты полинома степени **n**, который с наименьшей среднеквадратичной погрешностью аппроксимирует функцию **y(x)**, заданную в узловых точках векторами **x** и **y**.

Если **n+1** равно количеству узловых точек, то график полинома точно проходит через узловые точки с координатами **(x,y)** (получим **интерполяцию**).

```
>> x=(-3:0.2:3); y=sin(x);
```

```
>> p=polyfit(x,y,3)
```

```
p =
```

```
-0.0953 -0.0000 0.8651 0.0000
```