

СРЕДА ДЛЯ ВЫЧИСЛЕНИЙ И ВИЗУАЛИЗАЦИИ MATLAB

УЧЕБНОЕ ПОСОБИЕ,
подготовленное для сайта **www.exponenta.ru**

Дьяконов А.Г.

СОДЕРЖАНИЕ

Введение	3
§1. Знакомство с MatLab	4
§2. Работа с системой	6
§3. Теоретико-множественные операции	9
§4. Двоичные представления	10
§5. Порождение матриц и работа с ними	10
§6. Фокусы с размерностями	17
§7. Полезные функции	18
§8. Операции с файлами	19
§9. Разреженные матрицы	20
§10. Графика	22
§11. Циклы	30
§12. Логические массивы	32
§13. Оформление *.m-файлов	35
§14. Структуры	37
§15. Встроенные и анонимные функции	40
§16. Массивы ячеек	41
§17. Строки	44
§18. Интерполяция и полиномы	47
§19. Поэлементные операции	49
§20. О скорости...	52
§21. О точности и памяти...	61
§22. Примеры анимации, GUI	64
§23. Перестановки	65
§24. Символьные вычисления	66
§25. Задания для самостоятельной работы	68
§26. Как не надо программировать в системе MatLab	78
§27. Советы по оформлению m-файлов	80
§28. Аналоги системы MatLab	83
Литература, ссылки	84

Qu'on ne nous reproche donc plus le manque
de clarté, puisque nous en faisons profession.

Pascal

ВВЕДЕНИЕ

Данное учебное пособие специально подготовлено для конкурса методических разработок сайта [Exponenta.ru] и является частью книги

Дьяконов А.Г. Практикум на ЭВМ кафедры математических методов прогнозирования (системы WEKA, RapidMiner и MatLab): Учебное пособие. – М.: Издательский отдел факультета ВМиК МГУ им. М.В. Ломоносова; МАКС Пресс, 2010. – 133с.: ил. (ISBN 978-5-89407-432-0).

В принципе, это не совсем «классическая методическая разработка», однако изложенный ниже материал уже несколько лет успешно используется для обучения системе MatLab студентов кафедры математических методов прогнозирования факультета ВМК МГУ. Обучение ведётся на примерах, причём примеры подобраны таким образом, чтобы показать **тонкости использования всех основных команд**. Особое внимание уделено **эффективному программированию** в системе (см. §20 «О скорости...», §21 «О точности и памяти...»), а также некоторым программистским трюкам (см. §6 «Фокусы с размерностями»). Материал существенно отличается от содержания всех справочных и учебных пособий по подобным системам. Особую ценность представляет подборка **задач для самостоятельного решения**, за основу которой взяты собственные разработки, а также потрясающие задачи из пособия [Acklam] и блога [Loren].

В пособии описаны только основные команды системы (не входящие в многочисленные библиотеки). Кроме того, не описаны некоторые полезные дополнительные возможности, например составление отчётов. Автор считает, что это тема для отдельного пособия (которое будет в ближайшее время подготовлено), кроме того, она частично освещена, вместе с примерами решений реальных задач анализа данных в

Дьяконов А.Г. Практикум на ЭВМ кафедры математических методов прогнозирования (логические игры, обучение по прецедентам): Учебное пособие. – М.: Издательский отдел факультета ВМиК МГУ им. М.В. Ломоносова; МАКС Пресс, 2010. – 164с.: ил. (ISBN 978-5-89407-431-3)

Автор будет рад, если этот материал разместят на сайте [Exponenta.ru] для свободного доступа всем посетителям сайта. Обо всех неточностях и опечатках, найденных в тексте, можно сообщить по электронной почте djakonov@mail.ru.

Благодарности

Автор благодарен Юрию Ивановичу Журавлёву и Константину Владимировичу Рудакову, Леониду Моисеевичу Местецкому, Сергею Исаевичу Гурову, Константину Вячеславовичу Воронцову и Дмитрию Петровичу Ветрову за сотрудничество и советы.

Отдельное спасибо людям, которые своими ценными замечаниями помогли существенно улучшить учебное пособие: Вадиму Викторовичу Стрижову, Наталье Фёдоровне Дышкант и Алексею Валентиновичу Нефёдову. Данная книга получилась бы гораздо хуже без всех научных и учебно-методических ресурсов, перечисленных в главе «Литература, ссылки». Во время работы над учебным пособием автор был поддержан грантом РФФИ 10-07-00609.

СРЕДА ДЛЯ ВЫЧИСЛЕНИЙ И ВИЗУАЛИЗАЦИИ MATLAB

§1. Знакомство с MatLab

MatLab – это программный продукт компании «The MathWorks, Inc.» [MathWorks], предназначенный для инженерных, научных и прикладных вычислений, а также визуализации и анализа их результатов. MatLab предлагает мощный С-подобный язык¹ (который удобнее С++ и FORTRANa для реализации численных методов), прекрасную графику и большое число различных библиотек (**toolboxes**). Среда MatLab идеально подходит для решения задач машинного обучения, обработки сигналов и изображений и т.д. Пользователю предоставляется инструмент, с помощью которого можно загрузить и предобработать данные, запустить алгоритмы анализа, визуализировать результат, составить отчёт об экспериментах и получить исполняемый файл, который проводит нужные операции над этими данными. Слово MatLab означает **matrix laboratry** (матричная лаборатория). Все **вычисления** здесь **матричные**, и, в определенном смысле, только один (полезный) тип данных: матрица.

Так получилось, что самый лучший источник информации по MatLab – её справочная система, поэтому дальнейшее изложение построено в виде примеров, с помощью которых проще и быстрее изучить особенности среды MatLab. Мы остановимся только на основах программирования, принципах «правильного написания кода», эффективном использовании стандартных функций, оставив за рамками изложения подробное описание всевозможных библиотек.

При запуске системы появляются следующие окна:

1. Command Window (правое нижнее на рис. 1). В этом окне вводятся команды (после значка-приглашения `>>`). В нём же отображаются результаты выполнения. MatLab – это **интерпретатор**! Команды можно также записать в М-файл (текстовый файл *.m) в виде скрипта (последовательность команд) или функции (получает аргументы и выдаёт значения), тогда они будут запускаться при наборе в командном окне имени этого файла. Для создания такого файла вызовите редактор командой **edit** – появится окно редактора **Editor** (правое верхнее окно на рис. 1). Если нажать клавишу «стрелка вверх» после значка-приглашения `>>`, то отобразится предыдущая набранная команда (очень удобно, когда следующая команда незначительно отличается от предыдущей).

2. Command History (левое нижнее на рис. 1). В этом окне отображаются все команды, которые запускали на исполнение вводом в командном окне.

3. Workspace (левое верхнее на рис. 1). В этом окне отображаются все переменные, которые использует система в данный момент. Если набрать команду **a=1**, то в окне появится новая переменная: матрица **a** размера 1×1. Если набрать команду **b=2;**(точка с запятой), то новая переменная **b** появится в окне рабочего

¹ Этот язык называют также **MatLabом** или М-языком (M-code).

пространства, но в командном окне её результат не выводится. Это эффект действия «точки с запятой». Её используют, когда не требуется промежуточный вывод результатов, который может занять много времени или просто не нужен.

4. Help (окно помощи, вызываемое нажатием клавиши **F1**).

5. Current Directory (на рис. 1 «закрыто» окном Workspace, для активации необходимо щёлкнуть по вкладке с названием окна). Отображает **текущую директорию**. В этой директории система MatLab ищет файлы данных, которые Вы пытаетесь открыть и М-файлы, имена которых Вы набираете. Она также ищет их во всех каталогах, перечисленных в списке, доступном через меню **File / Set Path...** (его можно изменить вручную или пополнить командой **addpath**, см. ниже). В верхней части основного окна MatLab есть специальный компонент для выбора текущей директории (также для смены текущей директории можно использовать команду **cd**, см. ниже).

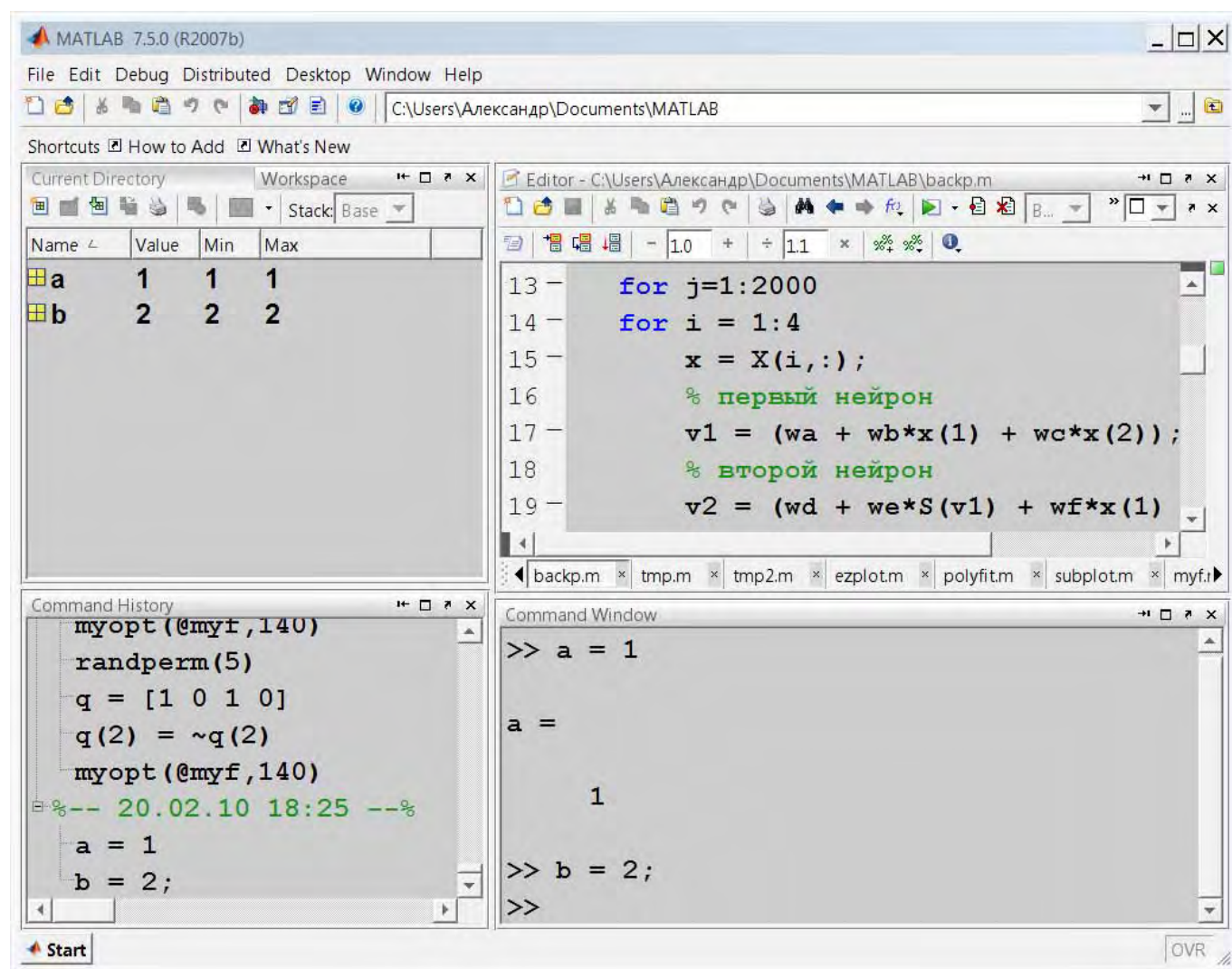


Рис. 1. Вид основного окна системы MatLab.

Назначение остальных окон будет ясно из контекста (например **Figure**) или файлов помощи (например **Profiler**).

При названии переменных **важен регистр**. Для начала работы можно набрать некоторые переменные, которым MatLab уже присвоил значения, хотя это не отображено в рабочей области:

i (мнимая единица),	eps (порог чувствительности),
j (мнимая единица),	realmax (наибольшее число),
pi (число пи),	realmin (наименьшее число),
Inf (бесконечность),	ans (результат последней операции),
NaN («не число», чтобы понять – наберите 0/0).	

Можно присваивать этим переменным новые значения, но тогда они теряют старые (попробуйте выполнить присваивание **pi=1**, команда **clear pi** вернёт исходное значение). Считается, что основное удобство при работе с системой MatLab – отсутствие необходимости определять переменные и распределять память, а также «краткость языка»: благодаря векторным (матричным) вычислениям и логическим массивам одна строка в этой среде заменяет несколько строчек кода на языке C++. Для этого, правда, необходимо научиться некоторым тонкостям работы с системой, например избегать по возможности оператора **for**.

Далее текст разбит на две колонки. В левой приводятся команды системы MatLab и результат их выполнения, в правой – необходимые пояснения¹. В каждой ячейке левого столбца таблицы приводятся команды, которые следует выполнять последовательно (следующие команды могут зависеть от результатов предыдущих).

§2. Работа с системой

<pre>>> help pi PI 3.1415926535897.... PI = 4*atan(1) = imag(log(-1)) = 3.1415926535897.... Reference page in Help browser doc pi >> help ans ANS Most recent answer. ANS is the variable created automatically when expressions are not assigned to anything else. ANSwer. Reference page in Help browser doc ans >> a = [1,2], b = a' a = 1 2</pre>	<p><i>Помощь. Попробуйте набрать sin(pi).</i></p> <p><i>Кстати, с помощью help elfun можно посмотреть список реализованных элементарных функций.</i></p> <p>lookfor cat – поиск в помощи слова «cat».</p> <p><i>Порождение двух векторов: вектор-строки и вектор-столбца. Штрих обозначает транспонирование. Элементы</i></p>
--	--

¹ Данный материал использовался на занятиях «Практикума на ЭВМ» для студентов 3 курса кафедры Математических методов прогнозирования ВМК МГУ. В электронной версии пособия можно копировать команды левой колонки, вставлять их в командное окно и запускать.

<pre> b = 1 2 >> a*b ans = 5 >> b*a ans = 1 2 2 4 >> a.*b' ans = 1 4 >> clear b >> whos Name Size Bytes Class Attributes a 1x2 16 double ans 1x2 16 double </pre>	<p>векторов заключаются в квадратные скобки!</p> <p>Скалярное произведение (строка на столбец).</p> <p>Внешнее произведение (столбец на строку).</p> <p>Поэлементное произведение. Все поэлементные операции «помечаются» точкой перед знаком операции.</p> <p>Удаление переменной b из памяти.</p> <p>Вывод всех переменных в памяти. Список переменных текущей рабочей области выводит команда who.</p>
<pre> >> path >> addpath c:\matlab7 >> cd C:\MATLAB7\work >> cd .. >> cd C:\MATLAB7 >> !dir </pre>	<p>Вывод путей доступа (определены в <i>pathdef.m</i>).</p> <p>Добавление нового пути</p> <p>Вывод текущего каталога (изменение его).</p> <p>Для вывода списка файлов в каталоге наберите what C:\MATLAB7\</p> <p>! или dos – вызов команды DOS.</p>
<pre> >> diary 1.txt >> diary off </pre>	<p>Писать в файл <i>1.txt</i> историю команд (вести дневник).</p> <p>Отключить ведение дневника.</p>
<pre> >> 1+1+1+... % переход +1+1 % комментарий ans = </pre>	<p>Три точки подряд – переход на следующую строку. Знак процента – комментарий.</p>

<p style="text-align: center;">5</p> <pre>>> format short; pi, format long; pi, format short e; pi, format long e; pi, format rat; pi ans = 3.1416 ans = 3.141592653589793 ans = 3.1416e+000 ans = 3.141592653589793e+000 ans = 355/113</pre>	<p>Форматы вывода чисел. См. помощь help format. Последний формат – приближение рациональной дробью. Команда format влияет только на формат вывода, а не на внутреннее представление.</p>
<pre>>> which plot C:\MATLAB\toolbox\matlab\graph2d\plot.bi >> edit >> quit >> clear a* >> clc</pre>	<p>Путь к файлу, в котором определена функция. В новых версиях выводится что-то типа built-in (C:\Program Files\MATLAB\R2007b\toolbox\matlab\graph2d\plot). Запуск редактора-отладчика. Завершить работу с Matlab. Удалить из рабочей области все переменные, которые начинаются на букву a. Очистить командное окно.</p>
<pre>>> [Inf, Inf, Inf, NaN, NaN, 0].*[Inf 0 -Inf Inf 0 Inf] ans = Inf NaN -Inf NaN NaN NaN</pre>	<p>Иллюстрация работы с «бесконечностью» (причём со знаком «плюс» или «минус») и «не числом». Обратите внимание, что элементы вектор-строки можно разделять пробелом, а можно – запятой. Попробуйте набрать -1/0 и 1/0.</p>
<pre>>> conj(i) ans = 0 - 1.0000i</pre>	<p>Пример вызова функции (комплексного сопряжения).</p>
<pre>>> demo</pre>	<p>Запуск демонстрации.</p>

§3. Теоретико-множественные операции

```
>> A = [1,3,5,3]; B = [4,2,2,3];
>> [A, B]
```

```
ans =
     1     3     5     3     4     2     2     3
```

```
>> union(A, B)
```

```
ans =
     1     2     3     4     5
```

```
>> setdiff(A, B)
```

```
ans =
     1     5
```

```
>> intersect(A, B)
```

```
ans =
     3
```

```
>> ismember(2, B)
```

```
ans =
     1
```

```
>> ismember(A, B)
```

```
ans =
     0     1     0     1
```

```
>> setxor(A, B)
```

```
ans =
     1     2     4     5
```

```
>> unique(B)
```

```
ans =
     2     3     4
```

```
>> issorted(ans)
```

```
ans =
     1
```

```
>> sort(B)
```

```
ans =
```

Задание двух множеств.
Конкатенация.

Объединение (обратите внимание на упорядоченность в ответе).

Разность.

Пересечение.

«Членство» (входит ли элемент 2 во множество B, 0 – нет, 1 – да, см. дальше «Логические массивы»).

«Членство». Особенность Matlab: применение функции сразу к массиву (множеству). Сравните с `sin([0,1,pi/2])`.

Симметрическая разность. Сравните с `union(setdiff(A, B), setdiff(B,A))` и `setdiff(union(A,B), intersect(A,B))`.

Уникальные элементы. Выдаёт массив (множество) без повторений. Сравните действие `union(A,B)` с `unique([A,B])`.

Отсортированность: является ли множество отсортированным.

Сортировка.

<pre> 2 2 3 4 >> [1 4 -2] + 1 ans = 2 5 -1 </pre>	<p>Увеличить все элементы множества на единицу. Обратите внимание, что допустимо также сложение $[1\ 4\ -2] + [1\ 1\ 1]$, но не $[1\ 4\ -2] + [1\ 1]$ (сложение матриц разных размеров).</p>
---	--

§4. Двоичные представления

<pre> >> a = bitget(19, 1:8) a = 1 1 0 0 1 0 0 0 >> b = bitget(24, 1:8) b = 0 0 0 1 1 0 0 0 >> bitand(a,b) ans = 0 0 0 0 1 0 0 0 >> dec2bin(1:8) ans = 0001 0010 0011 0100 0101 0110 0111 1000 </pre>	<p>Представление в двоичном виде (чисел 19 и 24 с помощью 8 бит).</p> <p>Конъюнкция (логическая операция И). Есть еще bitor (ИЛИ), bitxor (исключающее ИЛИ).</p> <p>Вывести все двоичные представления чисел от 1 до 8. При наборе команды на ЭВМ обратите внимание на тип ответа!</p>
--	--

§5. Порождение матриц и работа с ними

<pre> >> A = [1,2; 3,4] A = 1 2 3 4 >> A(1, 2) ans = 2 >> cat(3, A, A+1) </pre>	<p>Запятая (и пробел) является горизонтальной конкатенацией, «точка с запятой» – вертикальной.</p> <p>Обращение к элементу матрицы. Сначала указывается строка, потом столбец. Нумерация от единицы!</p> <p>Конкатенация по третьему (!)</p>
---	--

```

ans(:,:,1) =
    1    2
    3    4

ans(:,:,2) =
    2    3
    4    5

>> D = [zeros(3) 2*ones(3,4) ...
        3*eye(3,2), diag([4,5,1])]

D =
    0    0    0    2    2    2    2    3    0    4    0    0
    0    0    0    2    2    2    2    0    3    0    5    0
    0    0    0    2    2    2    2    0    0    0    0    1

>> D(1:2, 7:8)

ans =
    2    3
    2    0

>> D(:, 2*(1:6)) = []

D =
    0    0    2    2    0    0
    0    0    2    2    3    5
    0    0    2    2    0    0

>> D(:) '

ans =
    0  0  0  0  0  0  2  2  2  2  2  2  0  3  0  0  5  0

>> B = fix(10*rand([2 3]))

B =
    2    9    1
    5    9    9

```

направлению матрицы **A** и матрицы **A+1**. В **MatLab** матрицы многомерные! К матрице можно прибавлять скаляр или матрицу такого же размера.

zeros – матрица из нулей, **ones** – матрица из единиц, **eye** – единичная матрица, **diag** – диагональная матрица. Многоточие – перевод команды на другую строку¹ (команда пока не завершена).

Подматрица матрицы. **a:b** – вектор с элементами **a, a+1, ..., b**.

Удаление всех чётных столбцов. **[]** – пустая матрица. **:** (двоеточие) – все элементы в данном направлении. **2*a** – поэлементное умножение вектора.

Вывести все элементы матрицы. «Штрих» – транспонирование (чтобы выводилось в строку).

MatLab выводит элементы по столбцам! Попробуйте выполнить **D(7)**.

rand – случайная матрица (см. также **randn**), **fix** – отбросить дробную часть. См. также **round** (округление к ближайшему целому), **floor** (округление к наименьшему целому).

¹ По возможности будем переводить длинные команды на новую строку, но в некоторых случаях такой перевод будет неудобен и длинные команды MatLab'a займут несколько строк (с обычным переносом «по пробелам»). Такие «вынужденные переносы» будут легко идентифицироваться, тем более что начало каждой команды (кроме некоторых комментариев) помечаются символами **>>**.

```
>> [max(B); min(B); sum(B)]
```

```
ans =
```

```
    5    9    9
    2    9    1
    7   18   10
```

```
>> C = reshape(1:2:11, [2 3])
```

```
C =
```

```
    1    5    9
    3    7   11
```

```
>> min(C, B)
```

```
ans =
```

```
    1    5    1
    3    7    9
```

```
>> C(end, 1)
```

```
ans =
```

```
    3
```

```
>> B.*C
```

```
ans =
```

```
    2   45    9
   15   63   99
```

```
>> a = reshape(repmat(1:3,2,1), 1, 6)
```

```
a =
```

```
    1    1    2    2    3    3
```

Какие элементы могут быть в матрице **B**?

Максимальные и минимальные элементы в столбцах. Суммы столбцов. См. также **cumsum** (кумулятивная сумма), **prod** (произведение), **cumprod** (кумулятивное произведение), **sort** (сортировка матрицы), **mean** (ср. арифметическое), **median** (медиана).

reshape – изменение формы матрицы (форма меняется «по столбцам»). **a:b:c** – вектор из чисел **a**, **a+b**, ...

Команды **1:4**, **1:1:4**, **linspace(1,4,4)** эквивалентны (последняя «делит» отрезок **[1,4]** четырьмя точками). Попробуйте команду **logspace(1,4,4)**, которая «действует в логарифмической шкале».

Минимум (поэлементный) двух матриц.

Кстати, **min(C,[],2)'** и **min(C')** – минимальные элементы по строкам.

end – последняя позиция: **C(end,end)** – элемент в позиции (2,3), **C(end,:)** – последняя строка, **C(:,end)** – последний столбец.

Поэлементное произведение матриц. Точка – указание того, что операцию надо произвести поэлементно. **B.^2** – поэлементное возведение в квадрат.

Получение строки **1 1 2 2 3 3**. **repmat** – повтор матрицы. Например, **repmat([1,0;0,1],50,50)** – матрица типа шахматной доски. Попробуйте выполнить команду

```
>> A = [1 5 2; 3 2 1; 5 4 0];
>> [B I] = sort(A)
```

```
B =
     1     2     0
     3     4     1
     5     5     2
```

```
I =
     1     2     3
     2     3     2
     3     1     1
```

```
>> [B I] = sortrows(A, 2)
```

```
B =
     3     2     1
     5     4     0
     1     5     2
```

```
I =
     2
     3
     1
```

```
>> B = rand([1 2 3 5 1]);
```

```
>> size(B)
```

```
ans =
     1     2     3     5
```

```
>> size(permute(B, [2 1 4 3]))
```

```
ans =
     2     1     5     3
```

```
>> size(shiftdim(B, 1))
```

```
ans =
     2     3     5
```

```
>> [length(B), ndims(B), size(B,4)]
```

```
imagesc(repmat([1,0;0,1], 4, 4)).
```

Сортировка элементов столбцов(!) матрицы! Выводится результат сортировки и матрица перестановок индексов. Для вывода только результата наберите **B = sort(A)**. Для сортировки элементов строк – **sort(A,2)**. Для сортировки в обратном порядке – **B = sort(A,'descend')**.

Сортировать строки по возрастанию элементов второго столбца. Выводится перестановка номеров строк.

Порождение случайной матрицы размера 1×2×3×5×1.

Вывод её размера. Обратите внимание, что последняя (фиктивная!) размерность устранена. Для устранения всех фиктивных размерностей используйте **B = squeeze(B)**.

permute – перестановка размерностей.

shiftdim – сдвиг размерностей. Последняя фиктивная размерность всегда удаляется. Три способа транспонирования матрицы: **[shiftdim([1,2;3,4],1) , permute([1,2;3,4],[2,1]) , [1,2;3,4]']**.

length – длина вектора. Для матриц эквивалентно

```
ans =
      5      4      5
```

```
>> A = [], A(end+2) = 2
```

```
A =
      []
```

```
A =
      0      2
```

```
>> A = toeplitz([1 2 3])
```

```
A =
      1      2      3
      2      1      2
      3      2      1
```

```
>> A(:, 2:3) = A(:, [3 2])
```

```
A =
      1      3      2
      2      2      1
      3      1      2
```

```
>> rot90(A)
```

```
ans =
      2      1      2
      3      2      1
      1      2      3
```

```
>> x=A\[1 2 4]'
```

```
x =
      1.1250
     -0.3750
      0.5000
```

max(size(B)). ndims(B) – число размерностей, эквивалентно **length(size(B)). size(B,4)** – показать длину 4-й размерности.

Вектор (матрица) автоматически дополняется нулями. Команда **A(end+1) = val** добавляет к вектору **A** ещё один элемент. Присваивание **A = [val A]** добавляет его слева. Обратите внимание, что команды можно записывать в одной строке через запятую (или через точку с запятой – тогда не будет выведен промежуточный результат).

Порождение матрицы специального вида. См. также функции «галереи». Например, порождение циркулянтной матрицы **gallery('circul',1:3)** или случайной бинарной **gallery('rando',5)**.

Перестановка столбцов.

Поворот матрицы на 90 градусов.

Решение уравнения **A*x=[1 2 4]'**. Не надо думать! Просто формально поделить на матрицу **A**. Способ **x=inv(A)*[1 2 4]'** хуже, т.к. происходит инвертирование матрицы. Обратите внимание на направление деления, сравните **1/2** и **1\2**.

```
>> A*x - [1 2 4]'
```

```
ans =
    1.0e-015 *
           -0.2220
              0
              0
```

```
>> A^2
```

```
ans =
    13    11     9
     9    11     8
    11    13    11
```

```
>> sub2ind([3,2], [1,2,3], [1,1,2])
```

```
ans =
     1     2     6
```

Уравнение решено с погрешностью!

Возведение матрицы в квадрат. Сравните с поэлементным возведением **A.^2**.

Перевод многомерной нумерации в последовательную. Например, в матрице размера 3×2 элемент в позиции (2,1) соответствует второму элементу вектора всех элементов матрицы. См. также **ind2sub**.

```
>> A = reshape(1:9, 3, 3)
```

```
A =
     1     4     7
     2     5     8
     3     6     9
```

```
>> A(:, end) = [1 2 3]
```

```
A =
     1     4     1
     2     5     2
     3     6     3
```

```
>> A(end, :) = 5
```

```
A =
     1     4     1
     2     5     2
     5     5     5
```

```
>> A(1,2,1,1)
```

```
ans =
     4
```

```
>> A(1, 2, 2)
```

```
??? Index exceeds matrix dimensions.
```

При присваивании строка автоматически конвертируется в столбец (главное совпадение числа элементов). Можно писать также **A(:,end) = [1 2 3]'**, но нельзя **A(:,end) = [1 2]'**.

Константа присваивается сразу всем указанным элементам.

На единицы в несуществующих размерностях MatLab не обращает внимание.

На не-единицы обращает...

<pre>>> A(1:8) = rand([2 4]) A = 0.8147 0.9134 0.2785 0.9058 0.6324 0.5469 0.1270 0.0975 5.0000</pre>	<p>При присваивании вектору (!) происходит «подгонка» размеров. Нельзя сделать A(1:2,1:2) = rand([1 4]).</p>
<pre>>> A(2,2,2) = 3 A(:,:,1) = 0 0 0 0 A(:,:,2) = 0 0 0 3</pre>	<p>Создаётся матрица размера 2×2×2, все элементы которой, кроме элемента в позиции (2,2,2) равны нулю.</p>
<pre>>> accumarray([1 2 3 1 1 2]', [1 2 3 4 5 6]) ans = 10 8 3 >> accumarray([1 2 3 1 1 2]', [1 2 3 4 5 6], [], @prod) ans = 20 12 3</pre>	<p>Полезная функция accumarray. Формирует вектор, состоящий из суммы элементов, помеченных 1, суммы элементов, помеченных 2 и т.д.</p> <p>Аналогично с произведением. Обратите внимание: @prod – указатель на функцию prod.</p>
<pre>>> rank([A,b]) == rank(A)</pre>	<p>Определяет, существует ли решение системы A*x = b (матрица A может быть вырождена, т.е. её определитель det(A) равен нулю)? rank – ранг матрицы.</p>
<pre>>> [sum([]) sum([]+4) prod([])] ans = 0 0 1</pre>	<p>Обратите внимание на значение суммы и произведения элементов пустого множества, а также на то, что []+4 = [].</p>
<pre>>> A = [i 2*i; 3*i 4*i]; A' ans = 0 - 1.0000i 0 - 3.0000i 0 - 2.0000i 0 - 4.0000i >> A.' ans = 0 + 1.0000i 0 + 3.0000i 0 + 2.0000i 0 + 4.0000i</pre>	<p>Отличие транспонирования и «транспонирования с комплексным сопряжением».</p>

§6. Фокусы с размерностями

```
>> A = [1 1; 1 1]; % A = ones(2);
>> X = [A, 3*A; 2*A 4*A]
```

```
X =
     1     1     3     3
     1     1     3     3
     2     2     4     4
     2     2     4     4
```

```
>> Y = reshape(X, [2 2 2 2])
```

```
Y(:,:,1,1) =
         1         2
         1         2
```

```
Y(:,:,2,1) =
         1         2
         1         2
```

```
Y(:,:,1,2) =
         3         4
         3         4
```

```
Y(:,:,2,2) =
         3         4
         3         4
```

```
>> Y = permute(Y, [1 3 2 4]);
```

```
>> Y = reshape(Y, [2 2 4])
```

```
Y(:,:,1) =
         1         1
         1         1
```

```
Y(:,:,2) =
         2         2
         2         2
```

```
Y(:,:,3) =
         3         3
         3         3
```

```
Y(:,:,4) =
         4         4
         4         4
```

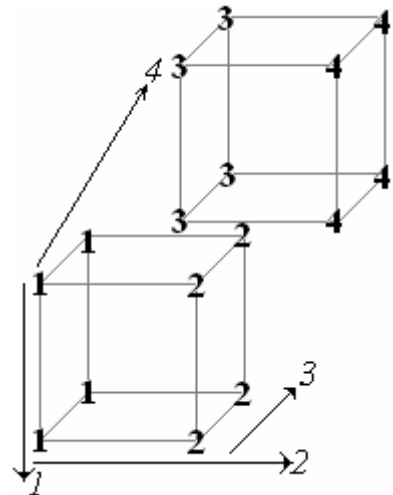
```
>> x = ones(1,2,1,0,0,1)
```

```
x =
Empty array: 1-by-2-by-1-by-0-by-0
```

```
>> squeeze(x)
```

Создаём матрицу, состоящую из четырёх блоков. Следующий код иллюстрирует, как продублировать эти блоки в третьем направлении, т.е. реализовать команду `cat(3, A, 2*A, 3*A, 4*A)`.

Превращаем матрицу в четырёхмерную.



Делаем перестановку размерностей и изменяем форму. Задача решена.

Попробуйте реализовать команду `cat(3, A, 3*A, 2*A, 4*A)` аналогичным способом.

Обратите внимание, что порождён пятимерный массив. Последняя размерность фиктивная, но предпоследняя (нулевая!) не является фиктивной.

<pre>ans = Empty array: 2-by-0-by-0</pre>	<p>Отбросили фиктивные (единичные) размерности.</p>
<pre>>> Y = reshape(X,[m size(X,1)/m n ... size(X,2)/n]); % сам поворот >> Y(:,:,,:) = Y(:,:,end:-1:1); % игра с размерностями >> Y = permute(Y,[1 4 3 2]); >> Y = reshape(Y, [m*size(X,2)/n ... n*size(X,1)/m]) % Если убрать вторую команду, % то получим обобщённую транспозицию</pre>	<p>Даны двумерные матрицы A, B, C, D, E, F, G, H, E одинаковых размеров [m n]. Матрица X имеет вид</p> <pre>X = [A B C; ... D E F; ... G H E].</pre> <p>Этот код осуществляет «внешний» матричный поворот [Acklam]:</p> <pre>[C F E; ... B E H; ... A D G].</pre>

§7. Полезные функции

<pre>>> x = 1; y = [1 2]; >> [x,y] = deal(y,x) x = 1 2 y = 1</pre> <pre>>> f = @(varargin) varargin{:}; >> [y x] = f(x,y) y = 1 2 x = 1</pre> <pre>>> [x y z] = deal(7) x = 7 y = 7 z = 7</pre>	<p>Функция deal позволяет «удобно» присваивать значения аргументам. Здесь показано, как поменять две переменные (даже разных типов). Заметим, что конструкция [x y] = [y x] не работает.</p> <p>Для этих целей можно использовать и анонимную функцию (см. также ниже).</p> <p>Ещё одно использование функции deal. Вместо записей</p> <pre>x = 7; y = 7; z = 7;</pre>
<pre>>> row = [1 1 2 3 3 2 3]'; >> col = [1 2 1 2 2 1 2]'; >> val = [1 2 3 4 5 6 7]'; >> [row, col, val] = find(accumarray(... [row, col], val, [], @sum, [], true)) row =</pre>	<p>Решение следующей задачи. Допустим, у нас есть три массива одной длины row, col, val. Элемент некоторой матрицы (потом мы увидим, что так реализуются в MatLab разреженные матрицы) с номерами (row(i), col(i))</p>

<pre> 1 2 1 3 col = 1 1 2 2 val = 1 9 2 16 </pre>	<p>имеет значение val(i). Если пара (row(i), col(i)) одна и та же для нескольких i, то элемент равен сумме соответствующих val(i). Требуется перевести данные в «сжатую форму», когда нет одинаковых пар (row(i), col(i)) для разных i. Например,</p> <pre> row = [1 1 2 3 3 2 3]'; col = [1 2 1 2 2 1 2]'; val = [1 2 3 4 5 6 7]'; </pre> <p style="text-align: center;">В</p> <pre> row = [1 1 2 3]'; col = [1 2 1 2]'; val = [1 2 9 16]'; </pre>
<pre> >> clear; exist('a') ans = 0 >> a = 10; exist('a') ans = 1 >> exist('pi') ans = 5 </pre>	<p>Существует ли переменная a?</p> <p>«Системная» константа pi.</p>
<pre> >> A = 1; >> L = 10; >> for i = 1:L >> A(i+1) = 2*A(i) + 1; >> end % «ЭКВИВАЛЕНТНЫЙ» КОД >> B = filter([1], [1 -2], ones(1,L+1)) B = 1 3 7 15 31 63 127 255 511 1023 2047 >> isequal(A, B) ans = 1 </pre>	<p>См. справку по функции filter, которая применяется при обработке сигналов, но может быть также полезна и для других задач.</p> <p>Сравнение на равенство.</p>

§8. Операции с файлами

<pre> >> save b.mat a >> load b.mat </pre>	<p>Сохранить массив с именем «a» в файл «b.mat».</p> <p>Загрузить данные из файла «b.mat».</p>
---	--

<pre>>> save b.txt a -ascii</pre>	<p>Сохранить в формате <i>ascii</i>. -tabs – разделение символов табуляцией, -v4 – создание файла для системы MATLAB4, -append – добавление в существующий файл.</p>								
<pre>>> fid = fopen('answer.txt', 'wt', 'n'); >> fprintf(fid, '%d\n', y); >> fclose(fid);</pre>	<p>На практике чаще всего используют такой способ сохранения (например, когда записывают в файл вектор-столбец ответов на задачу классификации). Поскольку команда save answer.txt y -ascii число 82 записывает как 8.2000000e+001.</p> <p>В MatLabe есть ещё следующие низкоуровневые функции для работы с файлами: fread, fscanf, fgetl, fseek.</p>								
<pre>>> f = fopen ('st.txt'); >> a = fscanf(f, '%d:%d %d/%d',... [4 100])'</pre> <p>a =</p> <table><tr><td>12</td><td>1</td><td>5</td><td>7</td></tr><tr><td>13</td><td>6</td><td>3</td><td>2</td></tr></table> <pre>>> fclose(f); >> g = fopen('st2.txt','w'); >> fprintf(g, ... '%2.2d:%2.2d %2.2f/%2.2d\n', a'); >> fclose(g);</pre>	12	1	5	7	13	6	3	2	<p>Чтение файла, в котором записано 12:01 5/7 13:06 3/2</p> <p>Большое значение «100» взято «на всякий случай».</p> <p>Попробуйте до закрытия файла повторить fscanf.</p> <p>Сохранение в файл в формате 12:01 5.00/07 13:06 3.00/02</p>
12	1	5	7						
13	6	3	2						
<pre>>> A = dlmread('1.txt', '+')</pre> <p>A =</p> <table><tr><td>1</td><td>2</td><td>3</td></tr></table>	1	2	3	<p>Загружаются данные из файла, который содержит единственную строку 1+2+3.</p> <p>Очень удобная функция для загрузки текстовых файлов (особенно с таблицами данных) textread.</p>					
1	2	3							

Загружать данные (бинарные, *ascii*-файлы, Excel-таблицы и т.д.) можно также выбрав в меню **File / Import Data**, что часто проще и удобнее.

§9. Разреженные матрицы

В анализе данных часто приходится иметь дело с матрицами больших размеров, почти все элементы которых нулевые. Для хранения таких матриц в

системе MatLab есть специальный тип – разреженные матрицы (sparse). При этом работа с разреженными матрицами происходит также как и с обычными.

```
>> A = fix(rand(3)*3)
```

```
A =
```

```
    2    1    0
    2    0    2
    2    0    0
```

```
>> S = sparse(A)
```

```
S =
```

```
    (1,1)    2
    (2,1)    2
    (3,1)    2
    (1,2)    1
    (2,3)    2
```

```
>> S(1,:)
```

```
ans =
```

```
    (1,1)    2
    (1,2)    1
```

```
A = speye([2 3])
```

```
A =
```

```
    (1,1)    1
    (2,2)    1
```

```
>> B = sparse([1 2 1], [2 3 1], ...
[4 7 2], 3, 4)
```

```
B =
```

```
    (1,1)    2
    (1,2)    4
    (2,3)    7
```

```
>> sparse(rand([2 2 2]))
```

```
??? Undefined function or method
'sparse' for input arguments of type
'double' and attributes 'full 3d real'.
```

```
>> spy(S)
```

Перевод матрицы в разреженную форму. Хранятся только ненулевые элементы. Обратный переход производится с помощью **full(S)**.

Первая строка разреженной матрицы.

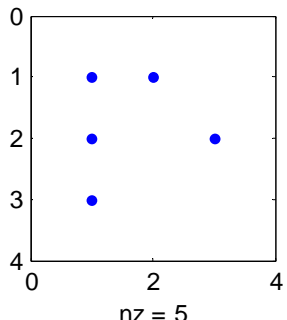
spones(A) – заменяет все ненулевые элементы на единицы, а **nnz(A)** – число ненулевых элементов.

Создание разреженной единичной матрицы. См. также **spdiags**, **sprand**, **sprandn**.

Создание разреженной матрицы. Перечисляются индексы элементов, их значения, указываются размеры матрицы. Попробуйте набрать без указания размеров матрицы: **sparse([1 2 1],[2 3 1],[4 7 2])**.

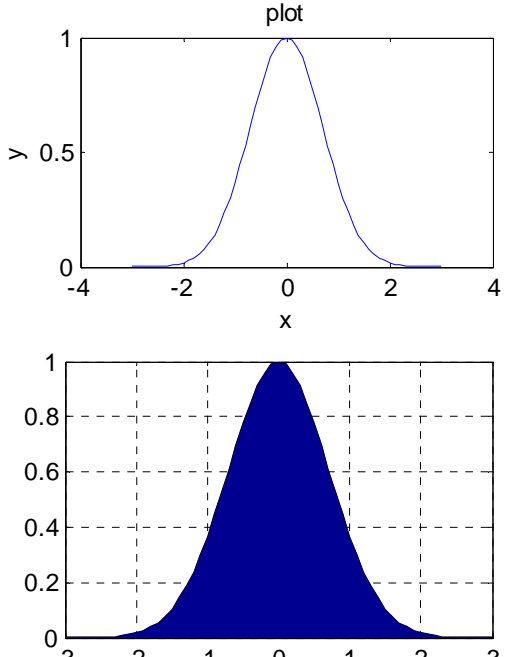
В MatLabе только двумерные разреженные матрицы!

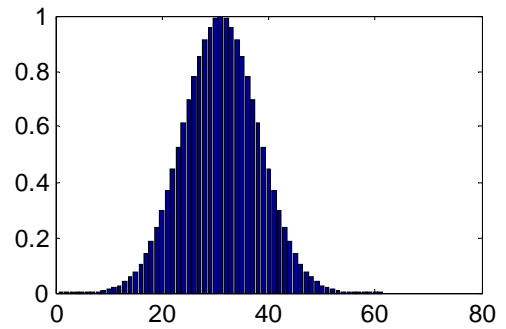
Визуализация матрицы:

	
<pre>>> S = spalloc(2,3,2); >> whos Name Size Bytes Class Attributes S 2x3 40 double sparse >> S(1,1) = 1; >> S(1,2) = 2; >> whos Name Size Bytes Class Attributes S 2x3 40 double sparse >> S(2,1) = 3; >> whos Name Size Bytes Class Attributes S 2x3 88 double sparse</pre>	<p>Выделить память для разреженной матрицы размера 2×3 с двумя ненулевыми элементами.</p> <p>Памяти хватает...</p> <p>«Лишний» элемент вызывает перераспределение памяти. Посмотрите на результат команд S = spalloc(2,3,3); whos. Лучше сразу знать, сколько памяти потребуется!</p>

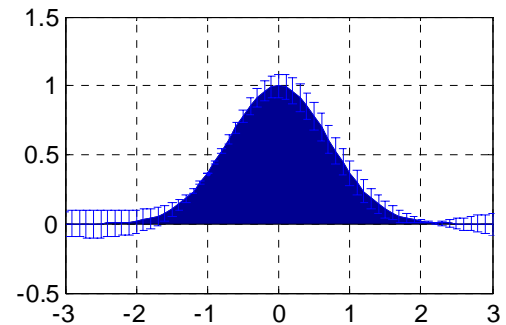
Если **s** – разреженная матрица, а **F** – обычная («полная»), то матрицы **s+s**, **s*s**, **s.*s**, **s.*F**, **inv(s)**, **diag(s)**, **max(s)**, **sum(s)** разреженные, а **s+F**, **s.*F** – полные.

§10. Графика

<pre>>> x = -3:0.1:3; >> y = exp(-x.*x); >> plot (x,y); >> title('plot'); % название >> xlabel('x'); % метки осей >> ylabel('y'); >> figure % создание нового окна >> area(x, y); % закрашенный график >> grid on; % нарисовать сетку >> figure(1); % вывод в первое >> bar(y) >> figure(2); >> hold on; % Рисовать в окне без удаления старого рисунка >> errorbar(x, y, sin(x+1)./10);</pre>	
--	--



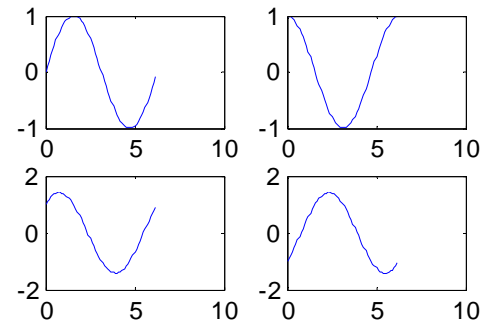
Пометки осей и название удалились!



Выполните эти команды последовательно. Попробуйте команду **plot** от двумерной матрицы.

```
>> clf % очистить окно вывода
>> t = (0:.1:2*pi)';
>> subplot(2, 2, 1)
>> plot(t, sin(t))
>> subplot(2, 2, 2)
>> plot(t, cos(t))
>> subplot(2, 2, 3)
>> plot(t, sin(t)+cos(t))
>> subplot(2, 2, 4)
>> plot(t, sin(t)-cos(t))
```

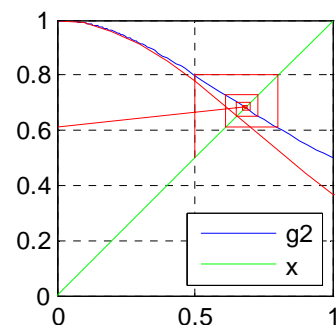
Построение нескольких графиков в одном окне.



Удалите команды **subplot**. Поставьте на их место команды **hold on**;

```
>> g2 = @(x) 1./(x.^2+1);
>> fplot(g2, [0 1]);
>> hold on
>> straightLine = @(x) x;
>> fplot(straightLine, [0 1], 'g')
>> legend('g2', 'x', ...
          'Location', 'SouthEast')
>> grid on
>> axis equal, axis([0 1 0 1])
>> x(1) = 0.5;
>> y(1) = x(1);
>> x(2) = x(1);
>> y(2) = g2(x(2));
>> for n = 3:2:21
>> x(n) = y(n-1);
```

Нахождение неподвижной точки [Loren].



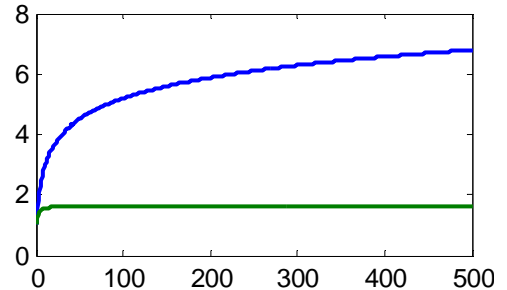
'r' - красный цвет (возможно

```
>> y(n) = y(n-1);
>> x(n+1) = x(n);
>> y(n+1) = g2(x(n+1));
>> end
>> plot(x,y,'r')
>> hold off
```

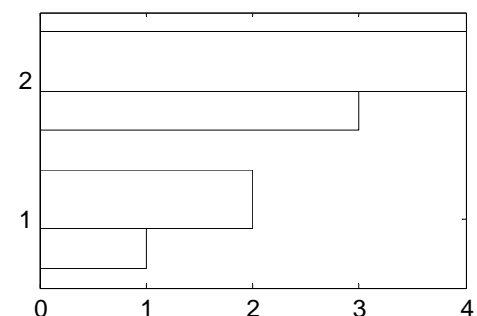
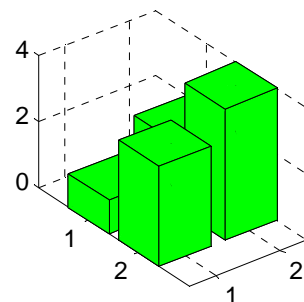
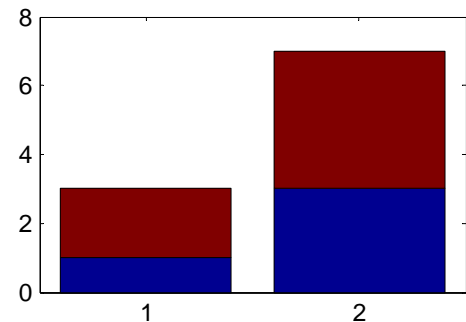
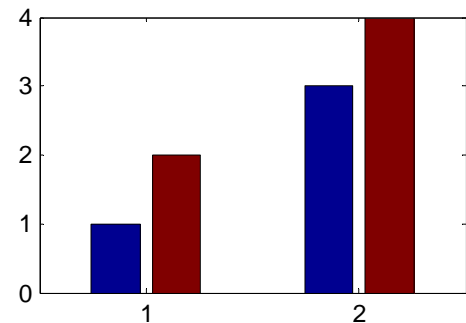
использование букв **b**, **g**, **r**,
c, **m**, **y**, **k**).

```
>> n = 1:500;
% кумулятивная сумма
>> s1 = cumsum(1./n);
>> s2 = cumsum(1./(n.*n));
>> plot([s1;s2]','LineWidth', 2)
```

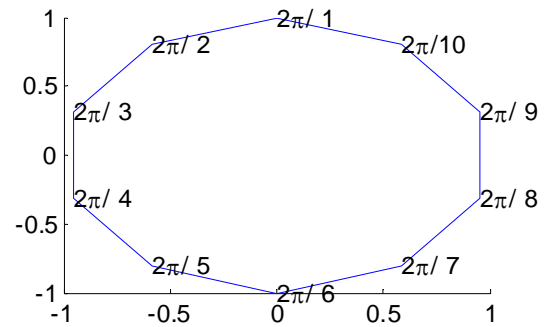
Иллюстрация сходимости и
расходимости рядов.



```
>> bar([1,2;3,4], 'grouped');
>> bar([1,2;3,4], 'stacked');
>> bar3([1,2;3,4], 'g');
>> barh([1,2;3,4], 1.5, 'w');
```

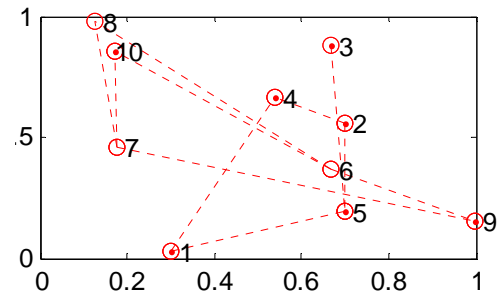


```
>> A = sin((0:10)*pi/5);
>> B = cos((0:10)*pi/5);
>> line(A, B); hold on;
>> text(A(2:11), B(2:11), ...
[repmat('2\pi/', 10, 1), ...
int2str(10-(0:9)')], 'FontSize', 10);
```



```
>> D = rand(10,2);
>> B = fix(triu(blkdiag(2*rand(5,5),...
2*rand(5,5))));
>> gplot(B,D,':og');
>> text(D(:,1),D(:,2),int2str((1:10)'));
```

Построение 2х случайных графов.



Кстати, **randn(m,n)** – случайная матрица (элементы распределены по нормальному закону) размера **m*n**.

```
>> saveas(gcf, 'myfig.fig')
```

Сохранение рисунка.

```
>> openfig myfig
```

Загрузка рисунка.

```
>> saveas(1, 'myfig.eps')
```

Сохранение **Figure 1** в формате **eps**.

```
>> set(gcf, 'paperpos', [0 0 3 2.25])
>> saveas(1, 'myfig.eps')
```

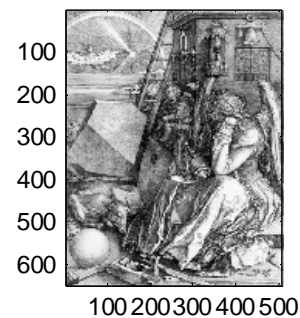
Попробуйте такой вариант сохранения. Посмотрите разницу. Попробуйте также **print -f -djpeg**.

```
>> load durer
>> whos
Name      Size      Bytes      Class
X          648x509   2638656    double array
ans        648x509   2638656    double array
caption    2x28       112        char array
map        128x3      3072       double array

Grand total is 660104 elements using
5280496 bytes
```

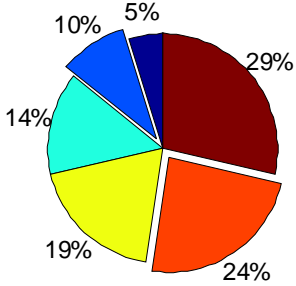
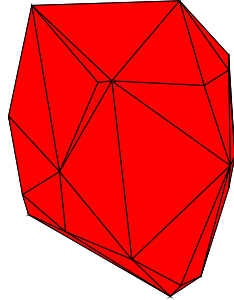
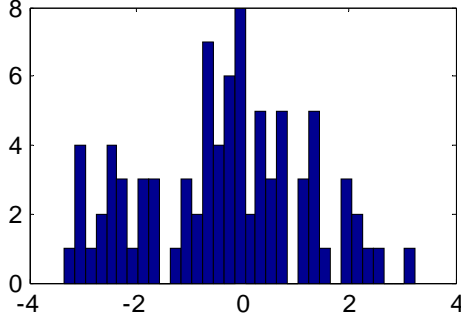
```
>> imagesc (X);
>> colormap(map);
>> axis image
```

Загрузка картинки.



```
>> pie([1 2 3 4 5 6], [0 1 0 0 1 0])
```

Секторная диаграмма («пирог»), с пометкой, какие сектора отделять.

	
<pre>>> A = randn(100, 3); >> C = convhulln(A); >> for i=1:size(C, 1) >> j = C(i, [1,2,3,1]); >> patch(A(j,1), A(j,2), A(j,3), 'r'); >> end; >> view(3), axis equal off tight vis3d; >> camzoom(1.5) >> camlight; lighting phong</pre>	<p>Выпуклая оболочка точек.</p> 
<pre>>> a = [randn([1 10])+2 randn([1 50])... randn([1 25])-2]; >> x = [min(a):0.2:max(a)]; >> hist(a, x) % Кстати, очень полезная функция... >> q = hist(a, x) q = 1 4 1 2 4 3 1 3 3 0 1 3 2 7 4 6 8 2 5 3 5 0 3 5 1 0 3 2 1 1 0 0 1</pre>	<p>Гистограмма.</p>  <p>Обратите внимание, что</p> <pre>>> q = hist([0 0.5 1 1.5 2],[0 1 2]) q = 2 2 1</pre> <p>Для изменения цвета наберите h = get(gca,'Children'); set(h,'FaceColor', 'r');</p>
<pre>>> [A, B] = meshgrid([1,2], [3,4]) A = 1 2 1 2 B = 3 3 4 4</pre>	<p>Действие функции meshgrid. Эта функция нужна для построения трёхмерных графиков (функций от двух переменных).</p>

```
>> x = 0:0.1:1;
>> y = linspace(0,1,11);
% или y=0:0.1:1
>> [X,Y] = meshgrid(x,y);
% или [Y,X] = ndgrid(y,x)
>> Z = sin(X*pi) + Y;
>> surf(x, y, Z);
```

Можно попробовать также:

```
>> mesh(Z);
>> meshz(Z);
>> surfl(z); shading interp;
>> colormap(pink);
>> contour(Z);
```

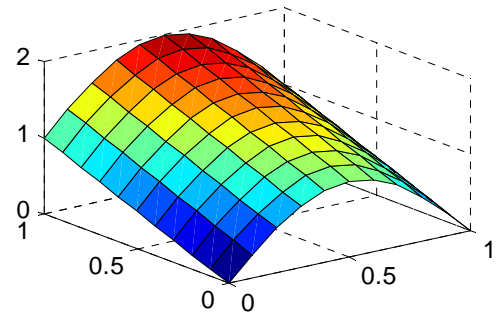
```
>> [x,y] = meshgrid(-2:0.2:2,-2:0.2:2);
>> z = x.*exp(-x.^2-y.^2);
>> [px,py] = gradient(z, 0.2, 0.2);
>> contour(z);
>> hold on;
>> quiver(px, py);
```

```
>> x = rand(10, 1); y = rand(10, 1);
>> plot(x, y, 'o');
>> hold on
>> xs = spline(1:10, x, 1:0.5:10);
>> ys = spline(1:10, y, 1:0.5:10);
>> plot(xs, ys, 'b');
>> xs = spline(1:10, x, 1:0.1:10);
>> ys = spline(1:10, y, 1:0.1:10);
>> plot(xs, ys, 'r');
```

```
>> x = -1:0.01:1;
>> q = plot(x, sin(x*10).*exp(x.*x));
% «СТИЛЬ» ЛИНИИ
>> set(q, 'LineStyle', ':');
% ТОЛЩИНА
>> set(q, 'LineWidth', 2);
```

```
>> set(q, 'Marker', 'o');
```

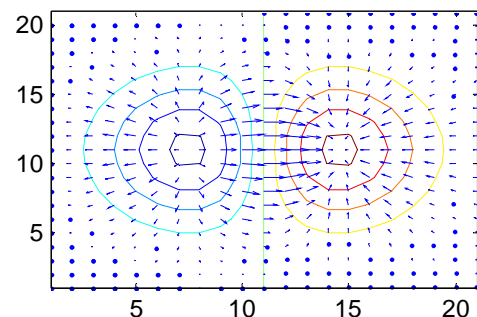
```
>> set(q, 'Color', 'red');
>> set(q, 'LineWidth', 3, ...
```



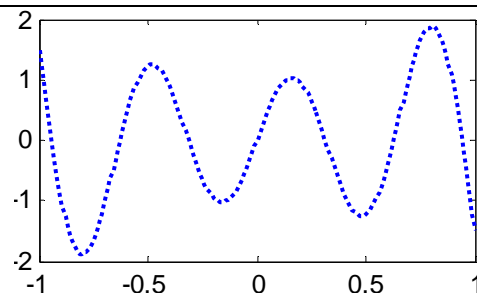
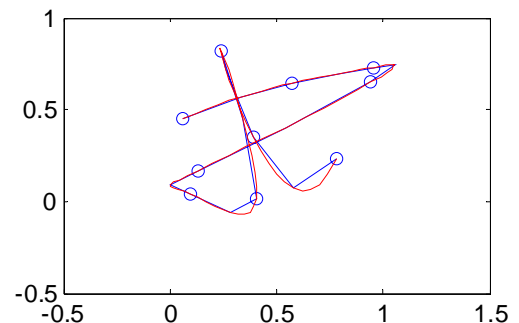
Параметры для **shading**:
interp, flat.

Параметры для **colormap**: **hsv, jet, cool, hot, gray, pink, cooper.**

Построение поля направления функции.



Сплайны.

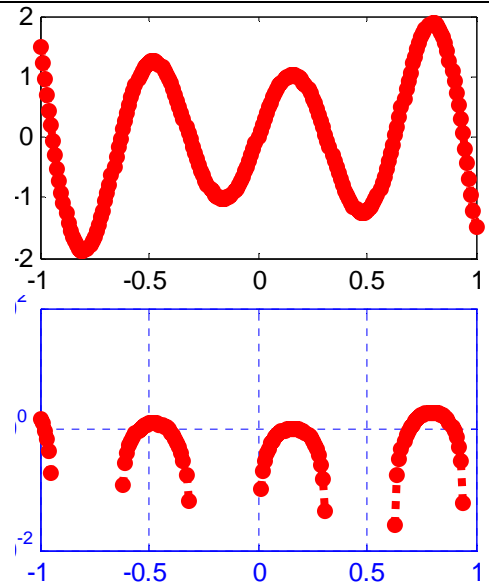


Параметры для **LineStyle**: **'-', ':", '---', '-.-'.**

Параметры для **Marker**: **'o', 'x', '+', '*', '.', 's', 'd', '^', 'v', 'h', 'p', '>', '<'.**

```
'MarkerSize', 3);
```

```
>> set(gca, 'XGrid', 'on', ...
           'YGrid', 'on');
>> set(gca, 'YScale', 'log');
Warning: Negative data ignored.
>> set(gca, 'XColor', 'blue', ...
           'YColor', 'blue');
```



Попробуйте применить команду **set(gca, 'fontsize', 10)** и команду **get(gca)**, чтобы узнать, что ещё можно менять...

```
>> x = -5:0.01:5;
>> for j = 0:0.01:20
>> plot(x, sin(x+j))
>> drawnow
>> end

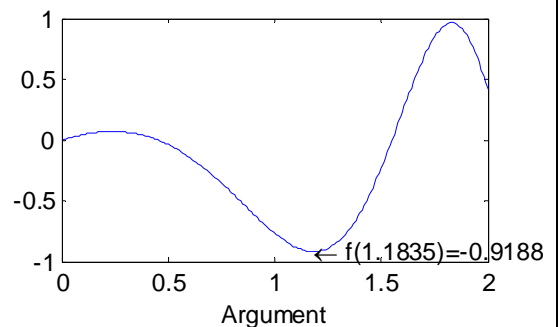
>> for j = 0:0.01:10
>> plot(x, cos(j)*exp(-x.*x));
>> set(gca, 'YLim', [-1 1]);
>> drawnow;
>> end
```

«Плывущие волны»

«Вращающаяся петля»

Задали диапазон изменения. Есть также функция **ylim**.

```
>> clf
>> f = @(t)cos(exp(t)).*sin(t);
>> T = 0:0.01:2;
>> [x, fm] = fminsearch(f,1);
>> s = strcat('\leftarrow f(', ...
             num2str(x), ')=' , num2str(fm));
>> plot(T, f(T))
>> text(x, fm, s, 'FontSize', 10);
>> xlabel('Argument')
```



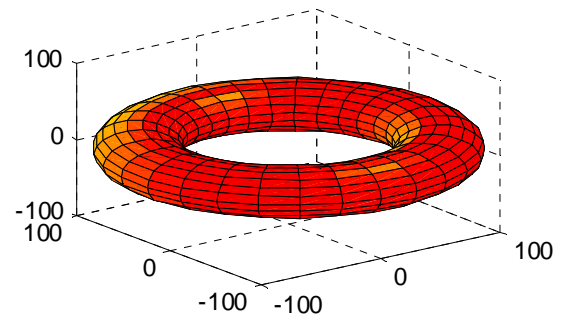
Пример минимизации функции. Показан пример использования анонимной функции (см. ниже).

```
>> t = 0:0.1:30;
>> x = t.*sin(t);
>> y = t.*cos(t);
>> clf
>> plot3(x,y,t);
>> axis off % убрать оси
```



```
>> [A B] = meshgrid(linspace(0,2*pi,20),
linspace(0,2*pi,30)); % перенос!

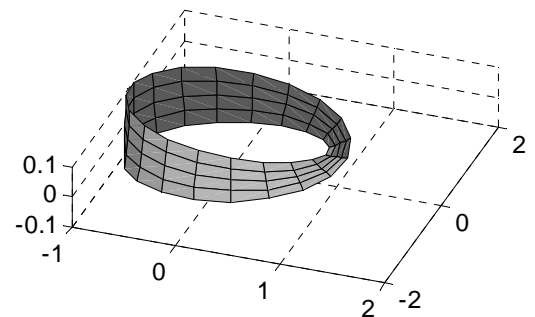
>> X = (100+30*cos(A)).*cos(B);
>> Y = (100+30*cos(A)).*sin(B);
>> Z = 30*sin(A);
>> surf(X, Y, Z);
>> axis([-100 100 -100 100 -100 100]);
>> colormap hot
```



Тор.

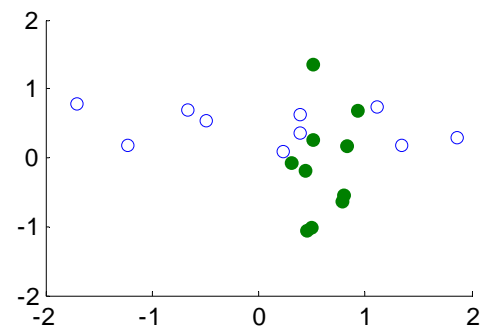
```
>> [A B] = meshgrid(linspace(0,2*pi,20),
linspace(-0.1,0.1,5)); % перенос!

>> X = cos(A)+B.*cos(A/2).*cos(A);
>> Y = sin(A)+B.*cos(A/2).*sin(A);
>> Z = B.*sin(A/2);
>> surf(X,Y,Z);
>> view(20,70);
>> colormap gray
```



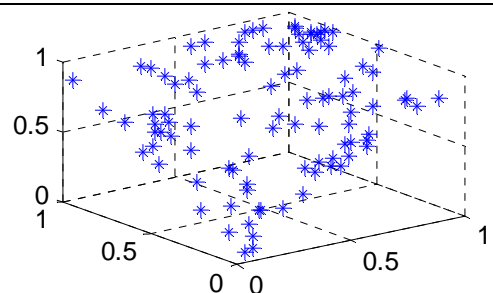
Лист Мебиуса.

```
>> x = randn([1 10]); y = rand([1 10]);
>> scatter(x, y);
>> hold on;
>> x = rand([1 10]); y = randn([1 10]);
>> scatter(x, y, 'filled');
```

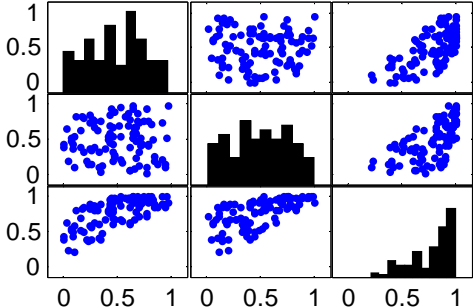


Визуализация данных (одна из самых полезных функций в анализе данных).

```
>> x = rand([1 100]);
>> y = rand([1 100]);
>> z = sin(x + y);
>> clf; scatter3(x,y,z);
```



Визуализация в трёхмерном пространстве. Попробуйте вращением картинки (Rotate

<pre>>> figure >> plotmatrix([x; y; z]')</pre>	<p>3D) убедиться, что данные лежат на «двумерной поверхности».</p> <p>Двумерные проекции множества точек.</p> 
<pre>>> ezcontour('x^2-y^2', [-1 1], [-1 1]) >> ezsurf('1/(1+x^2+2*y^2)', ... [-3 3], [-3 3]) >> ezplot('exp(3*sin(x)-cos(2*x))',... [0 4])</pre>	<p>Прямое построение графиков.</p>

Сохранить графику можно, выбрав **File/Save** в меню окна **Figure**, при этом сохранение производится в специальном формате ***.fig**. Можно также выбрать **File/Generate M-File**, тогда будет сгенерирован код, который рисует изображение, но в коде не присутствуют данные (их придётся внести вручную). Для использования построенных графиков в документах редактора Word используйте буфер: в окне **Figures** выберите **Edit/Copy Figure**, затем переключитесь в Word-документ и произведите вставку из буфера (**Paste**). Для использования графики в TeX используйте команду **print**.

§11. Циклы

<pre>>> A = 1; B = 2; >> if A > B % если >> '>' >> elseif A < B % иначе если >> '<' >> else % иначе >> '=' >> end % «если» завершается ans = <</pre>	<p>Пример условного оператора.</p> <p>В условном операторе следующие выражения эквивалентны:</p> <pre>if (a>b) (a>c), if or((a>b),(a>c)), if any([(a>b),(a>c)]).</pre>
<pre>>> f = [1 1]; >> for n = 3:10 % что пробегает n >> f(n) = f(n-1) + f(n-2); >> end % конец цикла >> f f = 1 1 2 3 5 8 13 21 34 55</pre>	<p>Вывод чисел Фибоначчи. Попробуйте усовершенствовать этот код (см. дальше об эффективном программировании в MatLab).</p> <p>Переменная может пробегать столбцы матрицы. См., например, for n = [1 2 3; 1</p>

<pre>>> n n = 10 >> S = 'qw12e3r4 56'; >> n = 0; >> for s = S >> if isletter(s) continue; >> elseif s==' ' break; >> end; % конец if >> n = n + str2num(s); >> end; % конец for</pre>	<p><code>1 1], n, end;</code> Кстати, допустим такой цикл:</p> <pre>for a=1:Inf, a, end;</pre> <p>но присваивание <code>a=1:Inf</code> без цикла <code>for</code> не удастся. Цикл <code>for a = [1:Inf], a, end;</code> недопустим, поскольку квадратные скобки сообщают MatLabу, что надо сначала вычислить выражение в них.</p> <p>После выполнения цикла переменная-счётчик остаётся в памяти (это способ проверки, прошёл ли цикл до конца, правда, с одной тонкостью... какой?).</p> <p>Использование continue и break. Что делает этот код? Попробуйте его усовершенствовать.</p>
<pre>>> x = 10; >> n = 0; >> while x > 1 >> x = x/2; n = n+1; >> if n > 50, break, end >> end</pre>	<p>Цикл while.</p>
<pre>>> switch isempty(A) >> case 0 >> 'непустой' >> case 1 >> 'пустой' >> otherwise >> '?' >> end ans = непустой >> x = 4; >> switch x >> case {1,2} >> disp('1<=x<=2'); >> case {3,4} >> disp('3<=x<=4'); >> end 3<=x<=4</pre>	<p>switch</p> <p>Пример проверки на пустоту массива.</p> <p>Допустимы и такие конструкции...</p>
<pre>>> E = [];</pre>	<p>Посмотрите на действия</p>

<pre>>> if E >> disp('Empty is true') >> else >> disp('Empty is false') >> end Empty is false >> true E ans = 1 >> true E ans = [] >> E true ??? Operands to the and && operators must be convertible to logical scalar values.</pre>	<p>условного оператора с пустым множеством.</p>
<pre>>> [1 1 0 0] [1 0 1 0] ans = 1 1 1 0 >> [1 1 0 0] [1 0 1 0] ??? Operands to the and && operators must be convertible to logical scalar values.</pre>	<p>Разница между операциями и .</p> <p>Получаем логический массив.</p>
<pre>>> x = NaN; >> if (x==x) disp('='); else disp('~'); end; ~</pre>	<p>В MatLabe (x==x) не всегда истина. Что будет при x = [] и x = Inf?</p>

В системе MatLab нет операторов безусловного перехода **goto** (как, впрочем, нет меток и номеров строк для перехода).

§12. Логические массивы

Логические массивы позволяют во многих случаях обходиться без операторов цикла. Для хорошего программирования в среде MatLab необходимо уметь применять этот тип данных (который отсутствует во многих других языках).

<pre>>> A = [1,2,3;4,5,6;7,8,9]; >> L = logical([1,0,0;0,1,0;1,1,0]); >> A(L) ' ans = 1 7 5 8 >> [i,j] = find(A>5 & A<9); >> [i,j]'</pre>	<p>Порождение логического массива.</p> <p>Вывод элементов, «помеченных» логическим массивом.</p> <p>Поиск индексов элементов, удовлетворяющих специальным</p>
---	---

```

ans =
     3     3     2
     1     2     3

>> find(isprime(1:20))

ans =
     2     3     5     7    11    13    17    19

>> X = [2,4,3,1];
>> X(X>2)

ans =
     4     3

>> X(X>2) = 2

X =
     2     2     2     1

>> whos L
  Name      Size      Bytes      Class
  L          3x3         9        logical array

Grand total is 9 elements using 9 bytes

>> L = L + 0;
>> whos L
  Name      Size      Bytes      Class
  L          3x3        72        double array

Grand total is 9 elements using 72 bytes

>> [[1,2];[3,4]]==[[1,3],[2,4]]

ans =
     1     1
     1     1

```

условиям. Сравните с командой **find(A>5 & A<9)** (индексы в последовательной нумерации). Для перевода последовательной нумерации в обычную можно использовать **[i,j] = ind2sub(size(A), find(A>5 & A<9))**.

Вывод всех простых чисел. **isprime** – проверка на простоту (результат – логический массив). **find** – вывод индексов ненулевых элементов. Заметьте, что **find(isprime(a))** не выводит все простые числа из **a**, а только их индексы. Для вывода чисел используйте **a(isprime(a))**.

Вывод всех элементов, которые больше 2. Аналогичный результат получается при **X(find(X>2))**.

Заменить на 2 все элементы, которые больше двух (это не самый эффективный вариант решения такой задачи, см. ниже).

Обратите внимание на изменение типа при прибавлении нуля к логическому массиву (для изменения типа в системе старше MatLab 5 рекомендуется использовать унарный плюс, см. ниже).

В результате сравнения порождается логический массив (обратите внимание на различный способ порождения двух матриц).

Для проверки на равенство

<pre>>> any(L) ans = 1 1 0 >> x = [1, Inf, 2, NaN]; >> x(finite(x)) ans = 1 2 >> x = 1:10; >> x((x<3) (x>6))&(x~=8)) ans = 1 2 7 9 10</pre>	<p>следует использовать isequal([[1,2]; [3,4]], [[1;3], [2;4]]). Кстати, эта функция может работать со многими аргументами, см. isequal(2, 1+1, 3-1).</p> <p>Есть ли в столбце ненулевой элемент (ответ для каждого столбца).</p> <p>all – проверка гипотезы, что в столбце все элементы ненулевые.</p> <p>Оставить в массиве только конечные числа.</p> <p>Запомните: <code>~=</code> – не равно (отличие от языка C), <code> </code> – логическое ИЛИ, <code>&</code> – логическое И.</p>
<pre>>> a = 5:6; >> a([0 1]) ??? Subscript indices must either be real positive integers or logicals. >> a(logical([0 1])) ans = 6</pre>	<p>Тривиальный факт, но часто на этом ошибаются... Для логической индексации надо использовать логический массив (тип <code>logical</code>)! Запись a([1 2]) допустима (это вывод первого и второго элемента), а запись a([0 1]) формально выводит нулевой(?) и первый.</p>
<pre>>> a = 1:3; >> b = a==2 b = 0 1 0 >> b(1) = 2 b = 1 1 0 >> b(3) = NaN ??? NaN's cannot be converted to logicals. >> find(b == true) ans =</pre>	<p>Присваивание элементам логического массива значений «не поддается логике». Обратите внимание, что в итоге получился логический массив. Кстати, (true==2) неверно!!!</p> <p>В MatLabe есть константы true и false. Правда, можно было бы писать find(b == 1).</p>

<pre> 1 2 >> A = true(1,3) % так тоже можно! A = 1 1 1 >> B = ones(1,3) B = 1 1 1 >> isequal(A,B) ans = 1 </pre>	<p>Иллюстрация, что MatLab не различает 1 и true.</p>
<pre> >> a = [-2 1 0 2 0 3]; >> +(a~=0) % первый способ ans = 1 1 0 1 0 1 >> +(~~a) % второй способ ans = 1 1 0 1 0 1 </pre>	<p>Решение задачи «сделать все ненулевые элементы матрицы единичными». Унарный «плюс» нужен для перевода в тип double!</p> <p>Этот способ часто бывает чуть быстрее. Для замены неединичных элементов на единичные (в полной матрице больших размеров) не делайте так: a(a~=0) = 1, лучше так: a = +(~~a). Кстати, аналогичный результат даёт команда a = double(~~a) (выше использовалась операция «унарный плюс»), а вот операция a = (~~a)+0 гораздо медленнее.</p>

§13. Оформление *.m-файлов

В m-файле можно сохранять последовательность MatLab-команд, которая будет выполняться, если в командной строке набрать имя соответствующего файла. «Запуск» скрипта (m-файла, содержащего последовательность команд) эквивалентен последовательному набору соответствующих команд. Если m-файл начинается с ключевого слова **function**, то в нём описана функция, для которой прописываются аргументы и значения (см. ниже).

Редактор для оформления m-файлов вызывается командой **edit**. Это обычный текстовый файл, поэтому может быть набран почти в любом текстовом редакторе. Файл следует сохранить в каталоге, который прописан в «путях доступа» (иначе скрипт/функция не будет вызываться по команде).

<pre> % QUADFORM решение квадратного уравнения % Это содержимое файла quadform.m function [x1,x2] = quadform(a,b,c) </pre>	<p>Вызов такой функции, записанной в m-файле, осуществляется командой:</p>
--	--

<pre>d = sqrt(b^2 - 4*a*c); x1 = (-b + d) / (2*a); x2 = (-b - d) / (2*a);</pre>	<pre>>> [r1,r2] = quadform(1,- 2,1) r1 = 1 r2 = 1</pre> <p>Обратите внимание на то, что переменная d локальная (её не будет в рабочей области после завершения работы функции). Функция имеет свою область переменных.</p> <p>Запомните! Переменные в функциях – локальные, а в скриптах – глобальные!</p>
<pre>% MPAR сравнение сигналов в 2х метриках function [p1 p2] = mpar(x, y) x = mmean(x); y = mmean(y); p1 = sum(abs(x-y)); p2 = sqrt(sum((x-y).^2)); % MMEAN вычесть среднее function x = mmean(x) x = x - mean(x); %{ код для иллюстрации вложенных функций %}</pre>	<p>Внутри функции может быть определена другая функция. Такие функции называются вложенными (nested). Вызов mmean([1 2 3]) «не работает»! Работают только вызовы [p1 p2] = mpar(x, y) и mpar(x, y) (если содержимое сохранено в файле mpar.m).</p> <p>Так выделяется целый блок комментария.</p>
<pre>>> f = memoizel(@sin) f = @memoizel/inner >> f(0) new ans = 0 >> f(0) ans = 0 >> f(1) new ans = 0.8415</pre>	<p>Сохраните в m-файл следующую функцию [Loren]:</p> <pre>function f = memoizel(F) x = []; y = []; f = @inner; % nested function function out = inner(in) ind = find(in == x); if isempty(ind) out = F(in); x(end+1) = in; y(end+1) = out; disp('new'); else out = y(ind); end end end</pre> <p>Теперь мы сообщаем, какой функцией у нас будет f (в</p>

<pre>>> f(0) ans = 0 >> s = functions(f) s = function: 'memoize1/inner' type: 'nested' file: [1x62 char] workspace: {[1x1 struct]} >> s.workspace{1} ans = f: @memoize1/inner F: @sin x: [0 1] y: [0 0.8415]</pre>	<p>данном случае sin). При обращении к ней вычисляется значение этой функции или, если к ней уже обращались с таким аргументом, значение просто берётся «из памяти».</p> <p>Здесь показана «память» функции.</p> <p>Попробуйте написать аналогичную функцию с памятью, к которой можно обращаться с вектором значений.</p>
<pre>>> profile on; % <<вызов функций>> >> profile report >> profile off</pre>	<p>Указывает время выполнения функций и время выполнения каждой строки(!) каждой вызываемой функции.</p>

Если в тексте m-файла встречается команда **keyboard**, то выполнение скрипта прекращается на время. Пользователь может вводить команды в командном окне. При вводе **return** продолжается выполнение m-файла. При использовании команды **return** в m-файле прекращается выполнение функции/скрипта. Зарезервированные имена переменных:

nargin – количество параметров, с которыми была вызвана функция,

nargout – количество выходных параметров, с которыми была вызвана функция.

Команда **global a b** сообщает функции о двух глобальных переменных: **a** и **b**. Заголовок **function myfunc(a, b, varargin)** описывает функцию с двумя аргументами **a**, **b** и, возможно, ещё какими-то, записанными в массиве ячеек¹ **varargin** (это ключевое слово). Для переменного числа выходных параметров используют имя массива ячеек **varargout**.

Многие функции системы MatLab (**sqrt**, **sin** и т.д.) являются встроенными (built in). Они очень эффективно выполняются, их выполнение нельзя прервать (нажатием Ctrl+C), и их код недоступен (это функции ядра).

§14. Структуры

<pre>>> a = f([1, 1]) a = Value: 1 Gradient: [2x1 double]</pre>	<p>Сохраните в файл f.m</p> <pre>function fx = f(x) fx.Value = (x(1)-1)^2+x(1)*x(2); fx.Gradient = [2*(x(1)-1) + x(2); x(1)];</pre>
--	--

¹ Про массивы ячеек см. ниже.

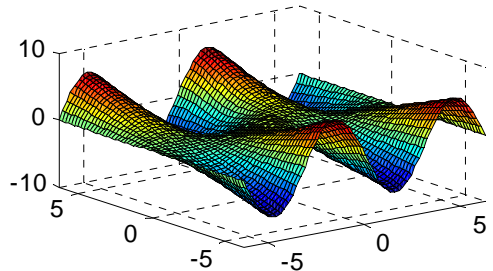
<pre>>> a.Gradient ans = 1 1 >> fieldnames(a) ans = 'Value' 'Gradient' >> isfield(a, {'Value', 'Trace', 'Gradient'}) ans = 1 0 1</pre>	<p>Значение поля.</p> <p>Вывод всех названий полей.</p> <p>Проверка, есть ли такие поля в структуре.</p>
<pre>>> student.name = 'Ivanov'; >> student.mark = 5; >> student(2).name = 'Petrov'; >> student(2).mark ans = [] >> student(3).name = 'Sidorov'; >> student(3).mark = 4; >> [student.mark] ans = 5 4 >> [student.mark] = deal(5, 4, 4) student = 1x3 struct array with fields: name mark % сразу несколько оценок >> [student(1:3).mark] = ... deal([5,4], [4,3,5], [4]) student = 1x3 struct array with fields:</pre>	<p>Массив структур с именами студентов и отметками.</p> <p>Поле mark у второго элемента уже существует! Можно было бы заполнить второй элемент так: student(2) = struct('name','Petrov','mark',[]).</p> <p>Массив всех оценок. Поскольку student.mark является массивом ячеек (см. ниже). Данный вызов работает и в случае, если student(1).mark = [5 4 5 5], но не в случае student(1).mark = [5 4 5 5]' (с транспонированием).</p> <p>Присваивание значений (проставили оценки сразу всем студентам). Попробуйте теперь вызвать [student.mark].</p> <p>Кстати, если сначала сделать присваивания student.name = 'Ivanov', student(2).name = 'Petrov', то присваивание student.mark = 5 некорректно. Верная запись - student(1).mark = 5.</p>

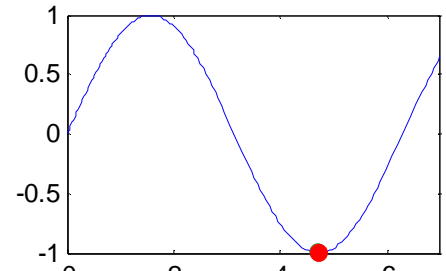
<pre> name mark % второй способ >> s = {[5,4], [4,3,5], [4]} s = [1x2 double] [1x2 double] [4] >> [student(1:3).mark] = s{1:3} student = 1x3 struct array with fields: name mark >> fields = fieldnames(s) fields = 'name' 'age' >> field1 = fields(1) field1 = 'name' snames = {student.(field1{:})} snames = 'Ivanov' 'Petrov' 'Sidorov' </pre>	<p>Присваивание не через deal. Сначала порожаем массив ячеек (см. ниже). Кстати, присваивание [student(1:3).mark] = {[5,4],[4,3],[4]} не получится.</p> <p>Дальше идёт иллюстрация одного способа доступа к полям... Сначала получили все имена полей.</p> <p>Выбрали первое.</p> <p>Вывели все значения этого поля... Команда student.(field1{:}) работает! Заметим, что команда snames = [student.(field1{:})] создаст строку 'IvanovPetrovSidorov'.</p>
<pre> >> s(1).x = 1; >> s(2).x = 2; >> a = struct2cell(s); % перевод в массив ячеек >> whos Name Size Bytes Class Attributes a 1x1x2 136 cell s 1x2 200 struct </pre>	<p>Обратите внимание, что «первая размерность структуры» – размерность имён полей.</p>
<pre> >> s1.name = 'Alex'; >> s1.age = 20; % попробуйте поменять местами эту строчку со следующей >> s1(2).name = 'Serg'; >> s1(2).age = 19; >> s2.mark = 5; >> s2(2).mark = 4; >> s = cell2struct([struct2cell(s1); struct2cell(s2)], [fieldnames(s1); </pre>	<p>Пример как объединять структуры (взято из [Loren]).</p> <p>Работает, если все имена полей уникальны!</p>

<pre>fieldnames(s2)],1) s = 1x2 struct array with fields: name age mark >> s(1) ans = name: 'Alex' age: 20 mark: 5</pre>	<p>Вывод первого элемента массива структур (теперь здесь три поля).</p>
<pre>>> field = 'f'; >> for i = 1:3; >> a.([field int2str(i)]) = i; end; >> a a = f1: 1 f2: 2 f3: 3</pre>	<p>Один из способов создания полей и присваивания им значений (динамические имена полей - Dynamic Field References).</p>

§15. Встроенные и анонимные функции

Для создания таких функций не нужен отдельный m-файл.

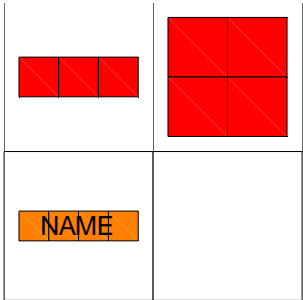
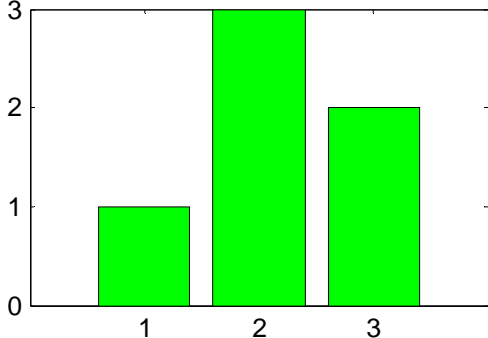
<pre>>> f = inline('sin(pi*x)+cos(pi*y)+z', 'z', 'x', 'y') f = Inline function: f(z,x,y) = sin(pi*x)+cos(pi*y)+z >> f(1,2,3) ans = -2.2204e-016 >> formula(f) ans = sin(pi*x)+cos(pi*y)+z</pre>	<p>Объявление встроенной функции.</p> <p>Обратите внимание: $x=2$, $y=3$, $z=1$! Попробуйте <code>f([1,2,3])</code>.</p> <p>Вывод формулы. Попробуйте также <code>argnames(f)</code> и <code>methods(f)</code>.</p>
<pre>>> f=@(x,y) (sin(x)*y+cos(y)/(x^2+1)); >> ezsurf(f);</pre>	<p>Анонимная функция. $(\sin(x) y + \cos(y) / (x^2 + 1))$</p> 

<pre>>> f = @(A,x) A(x) f = @(A,x)A(x) >> A = [1 2 3; 4 5 6; 7 8 9] A = 1 2 3 4 5 6 7 8 9 >> f(A,[1 2; 3 4]) ans = 1 4 7 2</pre>	<p>Создание анонимной функции, которая возвращает элементы матрицы A.</p> <p>Напомним, что MatLab нумерует элементы по столбцам. Подобная функция полезна для выражений типа f((triu(ones(3))+diag([1 2 3]))^2, 4), поскольку вызов ((triu(ones(3))+diag([1 2 3]))^2)(4) некорректен.</p>
<pre>>> f=@(x) (sin(x)+cos(x)-x); >> ezplot(f, [0 5]) >> grid on >> fsolve(f, 3) Optimization terminated: first-order optimality is less than options.TolFun. ans = 1.2587</pre>	<p>Решение нелинейного уравнения f(x)=0. Можно просто вызвать fsolve('sin(x)+cos(x)-x',3) (указывается начальное приближение).</p>
<pre>>> h = @sin; >> x = 0:0.01:7; >> fplot(h, [0 7]) % plot(x,h(x)) >> fminbnd(h, 0, 7) ans = 4.7124 >> hold on; >> scatter(ans, h(ans), 40,'filled')</pre>	<p>Пример использования указателя на функцию в задаче минимизации.</p>  <p>Попробуйте минимизировать свою функцию.</p>

§16. Массивы ячеек

Этот тип данных позволяет хранить в одном массиве элементы разных типов.

<pre>>> c = cell([2 2]) c = [] [] [] [] >> c{1, 1} = [1, 2, 3]; >> c{1, 2} = eye([2, 2]); >> c{2, 1} = 'NAME'; >> cellplot(c)</pre>	<p>Порождения массива ячеек.</p> <p>Заполнение его содержимым. Оно разнородное (разных типов)!</p> <p>Результат оператора cellplot:</p>
--	--

	
<pre>>> celldisp({datestr(now), now}) ans{1} = 26-Feb-2005 14:38:29 ans{2} = 7.3237e+005 >> datevec(datestr(now)) % г м час ч м с ans = 2005 2 26 14 38 45</pre>	<p>Вывод содержимого массива ячеек.</p> <p>Работа с датой и временем.</p>
<pre>>> T = cell([1,10]); >> T{1} = [1]; >> for i = 2:10; >> T{i} = [T{i-1},0]+[0,T{i-1}]; >> end;</pre>	<p>Задание треугольника Паскаля. Попробуйте выполнить celldisp(T).</p>
<pre>>> a = {1,[2 3],[4;5]} a = [1] [1x2 double] [2x1 double] >> a(1:2) ans = [1] [1x2 double] >> [a{1:2}] ans = 1 2 3 >> arg = {[1 3 2],'g'} arg = [1x3 double] 'g' >> bar(arg{:})</pre>	<p>Пример разницы в индексации массивов ячеек.</p> <p>Срез массива.</p> <p>Список значений. Посмотрите, что выдаёт команда a{1:2}.</p>  <p>Обратите внимание: гистограмма вывелась зелёным цветом.</p>

<pre>>> A = {[1],[2 3]}, 4} A = {1x2 cell} [4] >> A(end+1,:) = {1, 2} A = {1x2 cell} [4] [1] [2] >> A{2,2} = [1 2] A = {1x2 cell} [4] [1] [1x2 double] >> A(1,:) = [] A = [1] [1x2 double]</pre>	<p>Элементами массива ячеек могут быть массивы ячеек.</p> <p>Присваивание $A\{end+1,: \} = \{1,2\}$ невозможно!</p> <p>A здесь не пройдет присваивание $A(2,2) = [1\ 2]$ (с круглыми скобками). Можно сделать так: $A(2,2) = \{[1\ 2]\}.$</p> <p>Нельзя написать $A\{1,: \} = [].$</p>
<pre>>> X = [1 2 3; 4 5 6; 7 8 9]; >> a = {[1 3], ':'} a = [1x2 double] ':' >> a{:} ans = 1 3 ans = : >> X(a{:}) ans = 1 2 3 7 8 9</pre>	<p>Иллюстрация приёма, который используется для индексации массива.</p> <p>Вывод подматрицы матрицы X, определяемой массивом ячеек a.</p>
<pre>>> clear; a = 1; b = 2; X = [0 1]; >> vars = who' vars = 'X' 'a' 'b' >> class(vars) ans = cell</pre>	<p>Некоторые средства MatLab'a позволяют получить результат в виде массива ячеек. В этом примере получен массив имён всех переменных.</p>
<pre>>> c(1:4,1) = {'', 'Alex', 'Den', 'Serg'}; >> c(1:4,2) = {'MATH', 5, 4, 5}; >> c(1:4,3) = {'ECON', 5, 3, 4}</pre>	<p>Пример показывает, как из массива ячеек выделять вещественные подматрицы.</p>

<pre> c = '' 'MATH' 'ECON' 'Alex' [5] [5] 'Den' [4] [3] 'Serg' [5] [4] >> reshape([c{2:end,2:end}], size(c,1)- 1, size(c,2)-1) ans = 5 5 4 3 5 4 </pre>	
<pre> >> [b{1:3}] = deal(1, 2, 3) b = [1] [2] [3] >> [a{:}] = b{:} ??? The left hand side has a{:} inside brackets, which requires that a be defined, so that the number of expected results can be computed. >> [a{1:3}] = b{:} a = [1] [2] [3] </pre>	<p>Присваивания с помощью deal.</p> <p>Обратите внимание на эту ошибку!</p>

§17. Строки

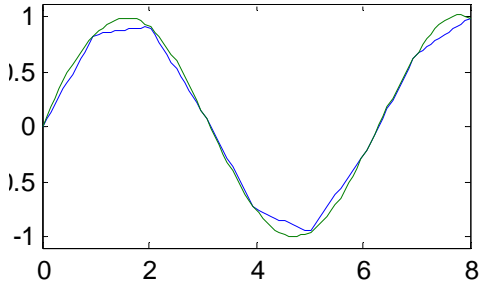
<pre> >> str = ['12 3', ' ', '4'] str = 12 3 4 >> double(str) ans = 49 50 32 51 32 52 >> char(ans) ans = 12 3 4 >> A = str2mat('1',upper('ab'),'456'7') A = 1 AB 456'7 >> findstr(A(3,:),''') </pre>	<p>Порождение строки.</p> <p>Преобразование в массив чисел.</p> <p>Обратное преобразование.</p> <p>Создание матрицы строк. Происходит «пополнение» пробелами (попробуйте double(A)). Обратите внимание на вставку апострофа.</p> <p>Определение позиции</p>
---	--

<pre>ans = 4 >> num2str(eval('1+1')) ans = 2 >> isletter('1a2bcd3') ans = 0 1 0 1 1 1 0 >> strmatch('ab', ['abc';'bca';'aaa']) ans = 1</pre>	<p>апострофа. Вообще, рекомендуется использовать функцию strfind, в которой важен порядок аргументов, но не функцию find(A(3,:)=='').</p> <p>Создание строки '2'. Запомните функцию eval! Она «исполняет строку». Попробуйте, например, eval('plot(sin(1:10))'). Некоторые задачи можно решать «компоновкой подходящей строки», которая затем передаётся функции eval.</p> <p>Определение позиций букв.</p> <p>Строки, которые начинаются с 'ab'.</p>
<pre>>> num2str(10.11111111111111) ans = 10.1111</pre>	<p>Будьте осторожны при использовании num2str, поскольку эта функция «обрубает числа»!</p>
<pre>>> findstr([0 1 2 0 0 3 4 1 1 2],[1 2]) ans = 2 9</pre>	<p>Строковые функции можно применять и для обработки «нестроковых» данных...</p>
<pre>>> strrep('Good boy and good girl', 'good', 'bad') ans = Good boy and bad girl >> strncmpi({'Good', 'boy', 'good', 'girl'}, 'GOOD', 4) ans = 1 0 1 0 >> strncmp({'Good', 'boy', 'bad', 'girl'}, 'b', 1) ans = 0 1 1 0</pre>	<p>Замена одной подстроки другой (можно и нужно использовать для удаления подстрок).</p> <p>Сравнение строк без учёта регистра.</p> <p>Какие строки начинаются на букву «b»?</p>
<pre>>> str = 'Value'; >> str + str</pre>	<p>При сложении строк результат</p>

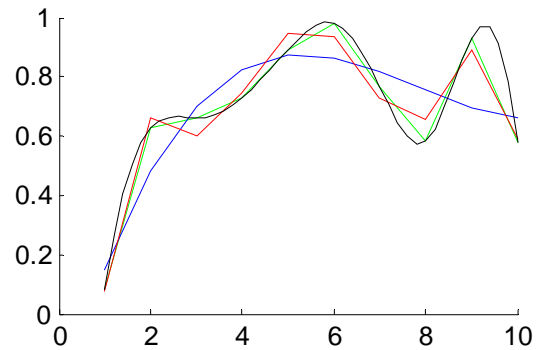
<pre>ans = 172 194 216 234 202 >> [str str] ans = ValueValue >> val = 3; >> sprintf('str = %s, val = %d or %f, pi = %2.3f', str, v, v, pi) ans = str = Value, val = 99 or 99.000000, pi = 3.142</pre>	<p>double!</p> <p>Конкатенацию следует производить так.</p> <p>sprintf – удобное средство для вывода информации на экран.</p>
<pre>>> X = [1 5 7]; >> n = 3; >> eval([' ' sprintf('A%d ',1:n) '] = ndgrid(' repmat('X, ',1,n-1) 'X ');]); >> eval(['A = [' sprintf('A%d(:) ',1:n) ']]); A = 1 1 1 5 1 1 7 1 1 1 5 1 5 5 1 7 5 1 1 7 1 5 7 1 7 7 1 1 1 5 5 1 5 7 1 5 1 5 5 5 5 5 7 5 5 1 7 5 5 7 5 7 7 5 1 1 7 5 1 7 7 1 7 1 5 7 5 5 7 7 5 7 1 7 7 5 7 7 7 7 7 >> G = repmat({X}, n, 1); >> [G2{1:n}] = ndgrid(G{:});</pre>	<p>Пример того, как с помощью функций sprintf и eval можно решать задачи любой сложности методом «формирования строки с нужной командой». Решается следующая задача. Для любого множества X и произвольного натурального n $n \geq 2$ надо построить матрицу, в которой по строкам перечислены все элементы множества X в декартовой степени n.</p> <p>Команда</p> <pre>['[' sprintf('A%d ',1:n) '] = ndgrid(' repmat('X, ',1,n-1) 'X ');']</pre> <p>формирует строку</p> <pre>'[A1 A2 A3] = ndgrid(X, X, X);'</pre> <p>которую исполняет команда eval. Аналогично, следующая команда формирует строку</p> <pre>'A = [A1(:) A2(:) A3(:)]'.</pre> <p>Второй способ решения задачи. Через массивы ячеек</p>

<pre>>> A = [G2{1:n}]; >> s = size(A); >> s = [s(1) length(X) n s(3:end)]; >> A = reshape(A, s); >> per = 1:ndims(A); per(3) = ndims(A); >> per(end) = 3; >> A = permute(A, per); >> A = reshape(A, [length(X).^n n])</pre>	<p>и фокусы с размерностями. Попробуйте понять, какой из этих двух способов эффективнее.</p>
<pre>>> ismember({'a','b','c'},{'a','m','n'}) ans = 1 0 0</pre>	<p>Для массива ячеек, наполненного строками, можно использовать ismember (попробуйте заменить одну из строк на число).</p>
<pre>>> S = []; >> for s = {'one','two','three'} >> S = [S ' ' s{:}] >> end S = one S = one two S = one two three</pre>	<p>Цикл может пробегать и массив ячеек.</p> <p>Обратите внимание, что s - ячейка! Попробуйте заменить эту строку на S = [S ' ' s].</p>
<pre>>> files = dir('*.mat'); >> for I = 1:length(files) >> thefile = files(i).name >> % действия с файлом >> end; >> for i = 1:9 >> thefile = sprintf('mat%d.mat', k); >> % другой способ >> thefile = ['mat' num2str(k) '.mat']; >> data = load(thefile); >> end;</pre>	<p>Так перебираются файлы в каталоге.</p> <p>Так перебираются файлы mat1.mat, mat2.mat и т.д.</p>

§18. Интерполяция и полиномы

<pre>>> x = 0:8; y = sin(x); >> xi = linspace(0, 8, 100); >> yiL = interp1(x, y, xi); >> yiC = interp1(x, y, xi, 'spline'); >> plot(xi, [yiL; yiC])</pre>	<p>Пример линейной интерполяции и интерполяции сплайнами.</p> 
---	--

```
>> x = 1:10;
>> y = rand(size(x));
% коэф. аппрокс. полинома 3-ей степени
>> p = polyfit(x, y, 3);
% вычисление его значений
>> f = polyval(p, x);
>> hold on;
>> plot(x, y, 'green');
>> plot(x, f, 'blue');
>> p = polyfit(x, y, 6);
>> f = polyval(p, x);
>> plot(x, f, 'red');
```

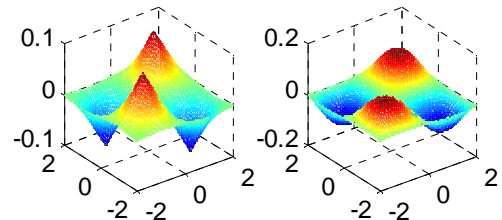


```
% аппрокс. кубическими сплайнами
>> g = spline(x, y, 1:0.2:10);
>> plot(1:0.2:10, g, 'black');
```

```
>> x = -2:2;
>> y = [-2:2]';
>> z = (sin(y).*exp(-
y.^2)).*(sin(x).*exp(-x.^2));
>> xi = linspace(-2,2,100);
>> yi = linspace(-2,2,100)';
>> ziL = interp2(x,y,z,xi,yi);
>> subplot(1,2,1), mesh(xi,yi,ziL)
>> title('Linear interpolation')
>> ziC = interp2(x,y,z,xi,yi,'spline');
>> subplot(1,2,2), mesh(xi,yi,ziC)
>> title('Cubic spline interpolation')
```

Пример двухмерной
интерполяции.

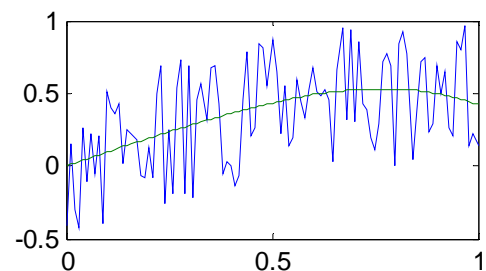
Linear interpolation Cubic spline interpolation



```
>> t = 0:0.01:1;
>> f = cos(t) + t + rand(size(t)) - 1.5;
>> t = t'; f = f';
>> F = [sin(t), sin(2*t), sin(3*t)];
>> a = F\f;
>> g = a(1)*F(:,1) + a(2)*F(:,2) + ...
      a(3)*F(:,3);
>> plot(t, [f,g])
```

Приближение функцией

$A \cdot \sin(t) + B \cdot \sin(2t) + C \cdot \sin(3t)$.



```
>> X = conv([1,-2,3], [1,-1])
```

```
X =
     1     -3     5     -3
```

```
>> roots(X)
```

```
ans =
  1.0000 + 1.4142i
  1.0000 - 1.4142i
  1.0000
```

Перемножение полиномов

Корни полинома.

```
>> polyder(X)
```

```
ans =
```

```
3 -6 5
```

Производная.

§19. Поэлементные операции

```
>> arrayfun(@abs, [-1 0;1 -2])
```

```
ans =
```

```
1 0
1 2
```

Поэлементная операция абсолютного значения над матрицей. Поэлементно можно выполнять любую функцию одного аргумента.

```
>> Y = repmat(X, length(p), 1).^repmat(p', 1, length(X))
```

```
Y =
```

```
1 1 1 1
2 3 4 5
4 9 16 25
8 27 64 125
```

Два способа построения матрицы Вандермонда. Элементы множества $X = 2:5$ возводятся во все степени $p = 0:3$.

```
>> Y2 = cell2mat(arrayfun(@(z)X.^z, p', 'uniformoutput', false))
```

```
Y2 =
```

```
1 1 1 1
2 3 4 5
4 9 16 25
8 27 64 125
```

```
>> A = fix(8*rand(3))
```

```
A =
```

```
1 5 2
5 6 1
3 7 5
```

```
>> A = bsxfun(@minus, A, min(A,[],2))
```

```
A =
```

```
0 4 1
4 5 0
0 4 2
```

```
>> A = bsxfun(@rdivide,A,max(A,[],2))
```

```
A =
```

```
0 1.0000 0.2500
0.8000 1.0000 0
0 1.0000 0.5000
```

Решение задачи нормировки. Построчно-линейным преобразованием перевести максимальный элемент каждой строки в 1, а минимальный – в 0. Функция **bsxfun** позволяет действовать построчно (не используя **repmat**).

Запомните:

```
>> bsxfun(@rdivide,[10 5;8 6],[5; 2])
```

```
ans =
```

```
2 1
4 3
```

```
>> bsxfun(@rdivide,[10 5;8 6],[5; 2])
```

```
ans =
```

```
2.0000 2.5000
1.6000 3.0000
```

```
>> [i,j] = meshgrid(1:1000, 1:1000);
```

Два решения задачи вычисления всевозможных длин гипотенуз

<pre>>> hypos = @(i,j) sqrt(i.^2+j.^2); >> tic; H = sqrt(i.^2+j.^2); toc Elapsed time is 0.141593 seconds. >> tic; H = bsxfun(hypos, i, j); toc Elapsed time is 0.125817 seconds.</pre>	<p>прямоугольных треугольников с длинами катетов от 1 до 1000.</p> <p>Функция tic делает «временную засечку», а toc сообщает, сколько времени прошло с момента этой засечки.</p> <p>Ещё одно полезное применение функции bsxfun, причём быстрое!</p>
<pre>>> S = {'djakonov@mail.ru', 'mmp@cs.msu.ru', 'mmp@cs.msu.su'} S = 'djakonov@mail.ru' 'mmp@cs.msu.ru' 'mmp@cs.msu.su' >> f = @(x) x(find(x=='@')+1:end) f = @(x)x(find(x=='@')+1:end) >> Sr = cellfun(f, S, 'UniformOutput', false) Sr = 'mail.ru' 'cs.msu.ru' 'cs.msu.su' >> any(strcmp(Sr, 'google.com')) ans = 0</pre>	<p>Решение задачи: оставить в перечне e-mail-адресов только соответствующие домены.</p> <p>Функция, которая «отбрасывает лишнее».</p> <p>Полезная функция cellfun (сделать что-то с каждой ячейкой, аналог функции arrayfun). В итоге получается массив ячеек. Заметим, что после выполнения cellfun(@length,S) получается обычный массив.</p> <p>Для определения, есть ли среди строк, содержащихся в массиве ячеек, заданная, можно использовать такую запись.</p>
<pre>>> S = sparse(fix(3*rand([2 3]))) S = (2,1) 2 (1,2) 1 (2,2) 1 (1,3) 2 (2,3) 2 >> f = spfun (@exp,S) f = (2,1) 7.3891 (1,2) 2.7183 (2,2) 2.7183 (1,3) 7.3891 (2,3) 7.3891</pre>	<p>Операция над всеми ненулевыми элементами разреженной матрицы. Очень полезна при работе с такими матрицами (хотя использование spfun иногда неэффективно, см. дальше).</p>

<pre>>> S.n1 = 'Name'; S.n2 = 'Mark' S = n1: 'Name' n2: 'Mark' >> structfun(@upper, S, 'UniformOutput', false) ans = n1: 'NAME' n2: 'MARK'</pre>	<p>Поэлементная функция для структур. В данном примере названия всех полей переводятся в верхний регистр.</p>
<pre>>> A = cell(2); >> [A{:}] = deal(0) A = [0] [0] [0] [0]</pre>	<p>Способ инициализации массива ячеек (автор Oliver Woodford [Loren]), кстати, более быстрый, чем <code>num2cell(zeros(2))</code>.</p>
<pre>>> funcs = repmat({'sin' 'cos' 'tan' 'log' 'exp'}, 1, 1000); % 1й способ >> tic; >> for f = funcs >> func = eval(['@' f{1}]); >> f = func(1.0); >> end >> toc Elapsed time is 0.467254 seconds. % 2й способ >> tic; >> for f = funcs >> func = str2func(f{1}); >> f = func(1.0); >> end >> toc Elapsed time is 0.142494 seconds. % изменение индексации во 2ом способе >> tic; >> for i = 1:numel(funcs) >> func = str2func(funcs{i}); >> f = func(1.0); >> end >> toc Elapsed time is 0.139898 seconds. >> tic; >> funcs = cellfun (@(x) ['@' x], funcs, 'UniformOutput', false); % переделка</pre>	<p>Есть перечень функций. В данном случае мы повторили 'sin', 'cos', 'tan', 'log', 'exp' 1000 раз. Рассмотрим рекомендации [Loren] по вычислению значений функций из перечня в фиксированной точке.</p> <p>Самый плохой (долгий) способ - вызывать <code>eval</code>.</p> <p>Если есть только перечень названий функций, то надо использовать <code>str2func</code>.</p> <p>Заметим, что при вызове <code>cellfun(@(x) ['@' x], {'sin' 'cos'}, 'UniformOutput', false)</code> ответ</p> <pre>ans = '@sin' '@cos'</pre> <p>Если есть возможность хранить не массив названий, а массив</p>

<p>данных и типов (получаем массив ячеек)</p> <pre>>> toc, tic; >> for i = 1:numel(funcs) >> f = funcs{i}(1.0); % и всё!!! >> end >> toc</pre> <p>Elapsed time is 0.111768 seconds. Elapsed time is 0.028202 seconds.</p>	<p>ячеек с указателями, то вычисления получаются сверхэффективными и эффективными.</p> <p>Переделка в массив ячеек занимает время. Зато вычисления потом происходят мгновенно!</p>
--	--

§20. О скорости...

	<p>Используйте функции, а не скрипты!</p> <ul style="list-style-type: none"> • Скрипты могут вызывать только функции (но не скрипты). • Функции сразу компилируются в память и хранятся там.
<p>% первый фрагмент</p> <pre>>> clear; tic; >> A = rand(100); >> y = ones(100, 1); >> dt = 0.001; >> for n = 1:(1/dt) >> y(:,n+1) = y(:,n) + dt*A*y(:,n); >> end; >> toc;</pre> <p>Elapsed time is 1.641000 seconds.</p> <p>% второй фрагмент</p> <pre>>> clear; tic; >> A = rand(100); >> y = ones(100, 1001); >> dt = 0.001; >> for n = 1:(1/dt) >> y(:,n+1) = y(:,n) + dt*A*y(:,n); >> end; >> toc</pre> <p>Elapsed time is 0.094000 seconds.</p> <p>>> A = int8(zeros(100)); % плохо >> A = zeros(100, 'int8'); % хорошо</p>	<p>Сразу выделяйте память!</p> <p>Сравните эти два фрагмента кода. Объясните разницу в скорости вычислений.</p> <p>Часто заранее неизвестно, сколько памяти нужно выделять. Тогда поступают примерно так...</p> <pre>X = zeros(1,n); for i = 1:n % какое-то присваивание X(i) = f(i); % выход по какому-то условию if b(i) break; end end % оставить только то, что надо b = b(1:ind);</pre> <p>Хуже писать в теле цикла X = [X f(i)] или X(end+1) = f(i).</p> <p>Не изменяйте размеры матриц в циклах!</p> <p>Сразу создавайте данные нужного типа!</p>


```
>> clear; tic; A = rand(1000);
>> for i = 2:size(A,1)
>> for j = 1:size(A,2)
>> A(i,j) = A(i-1,j) + A(i,j);
>> end
>> end; toc
```

Elapsed time is 2.797000 seconds.

```
>> clear; tic; A = rand(1000);
>> for i = 2:size(A,1)
>> A(i,:) = A(i-1,:) + A(i,:);
>> end; toc
```

Elapsed time is 0.156000 seconds.

```
>> clear; tic; A = rand(1000);
>> cumsum(A); toc
```

Elapsed time is 0.094000 seconds.

```
>> clear;
>> A = ones(1000,1000);
>> B = zeros(1000,1000);
>> C = zeros(1000,1000);
>> tic,
>> for n = 1:1000
>> for m = 1:1000
>> C(n,m) = A(n,m) + B(n,m);
>> end
>> end
>> toc
>> tic, D = A + B; toc
```

Elapsed time is 2.484000 seconds.

Elapsed time is 0.031000 seconds.

```
>> tic; s = 0;
>> for i = 1:10000
>> if (isprime(i)) s = s+i;
>> end;
>> end; s, toc;
```

s =

5736396

Elapsed time is 0.465790 seconds.

```
>> tic; sum( find( isprime(1:10000) ) ),
toc;
```

ans =

5736396

Elapsed time is 0.046846 seconds.

```
>> A = rand(1000,1000);
>> b = rand(1000,1);
```

Старайтесь использовать векторные вычисления!

Обратите внимание на время вычислений в этих трёх фрагментах кода.

Используйте встроенные функции!

*Ещё один пример. «Встроенное суммирование» эффективнее. Попробуйте убрать строчку **C = zeros(1000,1000);** (приготовьтесь ждать очень долго). Кстати, вычисления можно остановить нажатием **Ctrl+C**.*

Используйте маски (логические массивы)!

Задача: найти сумму простых чисел от 2 до 10000.

Решение с помощью векторизации и логических массивов.

Не вызывайте лишних функций!

```
>> tic, x1 = inv(A)*b; toc

Elapsed time is 1.516000 seconds.

>> tic, x2 = A\b; toc

Elapsed time is 0.578000 seconds.

>> D = diag(1:1000);
>> tic; D2 = inv(D); toc

Elapsed time is 0.962427 seconds.

>> tic; D3 = diag(1./diag(D)); toc

Elapsed time is 0.010591 seconds.

>> A = rand(1000);
>> tic; p1 = sum(1./eig(A)); toc

Elapsed time is 6.841646 seconds.

>> tic; p2 = trace(inv(A)); toc

Elapsed time is 1.125631 seconds.

>> sum(abs(p1 - p2))

ans =

    8.8818e-014
```

Не надо инвертировать матрицу, если этого можно избежать...

Инвертирование диагональной матрицы также лучше делать другим способом.

Правда, есть случаи, когда «традиционно плохие» функции (типа `inv`) могут быть очень полезны. Вот пример такой задачи (замечено Greg von Winckel в [Loren]).

Результаты «почти одинаковые» (из-за специфики операций с плавающей точкой, см. ниже).

```
>> A = rand(2000);
>> [X1 X2 X3 X4 X5] = deal(zeros(2000));
% сразу выделили память, чтобы на это не
% тратилось время
>> s = sum(A,2); % суммы всех строк

>> tic; X1 = diag(1./s)*A; toc

Elapsed time is 6.822780 seconds.

>> tic; X2 = A./repmat(s,1,size(A,2));
toc

Elapsed time is 1.875061 seconds.

>> tic; X3 = A.*repmat(1./s, 1,
size(A,2)); toc

Elapsed time is 0.130031 seconds.

>> tic; X4 = bsxfun(@rdivide,A,s); toc

Elapsed time is 0.278485 seconds.
```

Экспериментируйте!

Попробуйте несколько вариантов решения одной задачи и выберите оптимальный. Здесь представлено решение типичной задачи нормировки матрицы: каждый элемент надо разделить на сумму элементов в строке. Для чистоты эксперимента следует повторить присваивания матрицам **X1**, **X2** и т.д. в разном порядке. Заметим, что специально сразу выделили память под эти матрицы, чтобы на выделение не тратилось время.

Ещё один способ сделать нормировку!

```
>> tic; X5 = sparse(diag(1./s))*A; toc
Elapsed time is 0.238213 seconds.
```

```
>> tic; X6 = diag(sparse(1./s))*A; toc
Elapsed time is 0.187723 seconds.
```

```
>> [sum(X1(:)~=X2(:)) sum(X1(:)~=X3(:))
sum(X2(:)~=X3(:))]
```

```
ans =
    1439405         0    1439405
```

```
>> [sum(abs(X1(:)-X2(:))) sum(abs(X1(:)-
X3(:))) sum(abs(X2(:)-X3(:)))]
```

```
ans =
    1.0e-012 *
    0.1073     0    0.1073
```

```
>> clear
>> A = sparse(ceil(2000*rand([1
100000])), ceil(2000*rand([1 100000])),
rand([1 100000]), 2000, 2000);
>> s = sum(A,2);
>> tic; X1 = diag(1./s)*A; toc
```

```
Elapsed time is 0.894294 seconds.
```

```
>> tic; X2 = A./repmat(s,1,size(A,2));
toc
```

```
Elapsed time is 1.241347 seconds.
```

```
>> tic; X2 = A./repmat(full(s), 1,
size(A,2)); toc
```

```
Elapsed time is 0.336742 seconds.
```

```
>> tic; X3 =
A.*repmat(1./s,1,size(A,2)); toc
```

```
Elapsed time is 0.181846 seconds.
```

```
>> tic; X3 =
A.*repmat(full(1./s),1,size(A,2)); toc
```

```
Elapsed time is 0.185420 seconds.
```

```
>> tic; X4 = bsxfun(@rdivide,A,s); toc
```

Первый способ можно СУЩЕСТВЕННО улучшить, если умножать на разреженную диагональную матрицу.

Ещё большее улучшение получается, если диагональную матрицу сразу породить разреженной.

Вторая матрица отличается от первых двух!

На самом деле, это специфика арифметики с плавающей точкой! Это надо учитывать.

Для разреженных матриц действуют другие законы!

Предыдущий пример для разреженной матрицы. «Самый долгий» первый способ теперь становится быстрее (для сильно разреженной матрицы он может стать самым быстрым).

Обратите внимание на сокращение времени при приведении матрицы к полной!

Для разных операций действуют разные законы! Попробуйте повторить все эксперименты, не нормируя данные, а центрируя:
bsxfun(@minus,A,mean(A,2)).

Elapsed time is 0.298258 seconds.

```
>> tic; X5 = diag(sparse(1./s))*A; toc
```

Elapsed time is 0.014129 seconds.

```
>> clear; S = sparse(fix(3*rand(1000)-1));
```

```
>> tic; abs(S); toc
```

Elapsed time is 0.026724 seconds.

```
>> tic; spfun(@abs,S); toc
```

Elapsed time is 0.069756 seconds.

```
>> tic; S+1; toc
```

Elapsed time is 0.036316 seconds.

```
>> tic; 2*S; toc
```

Elapsed time is 0.009565 seconds.

А вот самый быстрый метод! Правильно всё-таки делать так:

```
X1 = sparse(1:size(A,1),  
1:size(A,1), (1./s))*A;
```

Новая серия экспериментов с разреженной матрицей...

Встроенные «универсальные» функции часто эффективнее поэлементной реализации.

*Сложение выполняется медленнее умножения! Всё дело в том, что **если** **меняется множество ненулевых элементов в разреженной матрице, то операция выполняется дольше!***

```
>> A = nan(2000);
```

```
>> B = ones(2000);
```

```
>> C = zeros(2000);
```

```
>> tic; A+B; toc
```

Elapsed time is 1.160856 seconds.

```
>> tic; A+C; toc
```

Elapsed time is 1.160238 seconds.

```
>> tic; B+C; toc
```

Elapsed time is 0.078591 seconds.

```
>> tic; B*C; toc
```

Elapsed time is 6.451293 seconds.

```
>> tic; A*C; toc
```

Elapsed time is 6.449248 seconds.

Некоторые данные «тормозят» работу MatLab.

В примере показано, что операции с матрицей, заполненной NaNами происходят дольше.

Интересно, что для умножения ощутимой разницы нет!

```
>> x = round(rand(1, 1e7));
>> tic; y = 1./x; toc % половина делении
на ноль
```

```
Elapsed time is 1.540780 seconds.
```

```
>> x = x + 1; % теперь делений на ноль
не будет
>> tic; y = 1./x; toc
```

```
Elapsed time is 0.329759 seconds.
```

```
>> str = char(fix(26*rand([1
10000000]))+'a'); % создали случайную
строку
```

```
>> tic; f1 = findstr(str,'z'); toc
```

```
Elapsed time is 0.117200 seconds.
```

```
% рекомендуется такой способ
>> tic; f2 = strfind(str,'z'); toc
```

```
Elapsed time is 0.117126 seconds.
```

```
>> tic; f3 = find(str=='z'); toc
```

```
Elapsed time is 0.312151 seconds.
```

```
>> str = fix(26*rand([1 10000000]));
```

```
>> tic; f1 = findstr(str, [0]); toc
```

```
Elapsed time is 0.085916 seconds.
```

```
>> tic; f2 = strfind(str, [0]); toc
```

```
Elapsed time is 0.084098 seconds.
```

```
>> tic; f3 = find(str==0); toc
```

```
Elapsed time is 0.173870 seconds.
```

```
>> clear
```

```
>> str = char(fix(26*rand([1
```

«Формирование» NaNов замедляет вычисления. Часто «выручает» добавления к вектору **eps**, чтобы избежать делений на ноль.

Есть очень быстрые способы вычислений! Надо просто их знать... Сначала рассмотрим пример поиска в очень большой строке буквы «z».

Это самый лучший способ!

А вот «интуитивно лучший» оказался медленным! Парадокс, но не всегда логические массивы хороши.

Рассмотрим такую же задачу, но с другими типами данных: среди массива чисел (от 0 до 25) найти все нулевые элементы.

Для «числовой задачи» лучшим оказался способ, который использует строковые функции! Отметим, что отсюда не следует, что все элементы надо искать как строки (необходимо помнить о специфике арифметики и представления данных в MatLabе, см. ниже):

```
>> strfind(0.01:0.01:0.07,
[0.06])
```

```
ans =
```

```
    []
```

Теперь рассмотрим удаление буквы из строки.

<pre>10000000]))+'a'); >> str1 = str; >> tic; str1 = str1(str1~='z'); toc Elapsed time is 0.406539 seconds. >> str2 = str; >> tic; str2(strfind(str2, 'z')) = []; toc Elapsed time is 0.455166 seconds. >> str3 = str; >> tic; str3 = strrep(str3,'z',''); toc Elapsed time is 0.234637 seconds.</pre>	<p>Здесь уже применение логического массива лучше strfind.</p> <p>Но самый лучший метод опять строковый!!!</p>
<pre>>> Z = rand([1 10000000]); % Замещение нужными элементами >> A = Z; >> tic; A = A(A>0.5); toc Elapsed time is 0.598856 seconds. % Удаление лишних >> A = Z; >> tic; A(A<=0.5) = []; toc Elapsed time is 1.010796 seconds. % Плохой способ замещения >> A = Z; tic; A = A(find(A>0.5)); toc Elapsed time is 0.873953 seconds. % Создание нового массива (ещё быстрее!) >> A = Z; >> tic; B = A(A>0.5); toc Elapsed time is 0.556692 seconds. % Предварительная операция с основным массивом >> clear B >> A = Z; >> A(1) = A(1)+sin(0); >> tic; A = A(A>0.5); toc Elapsed time is 0.567282 seconds. % Пример того, как не надо делать >> A = Z;</pre>	<p>Даже простейшие операции допускают несколько способов реализации. Выбирайте лучший!</p> <p>Здесь показано несколько способов удаления элементов массива.</p> <p>Ещё раз... Не вызывайте лишних функций! В данном случае find. Вообще, вместо a(find(a>0.5)) = 1 пишите a(a>0.5) = 1 (типичная ошибка). Попробуйте сравнить</p> <pre>>> A = Z; tic; L = find(A>0.5); A(L) = 1; toc >> A = Z; tic; L = A>0.5; A(L) = 1; toc</pre> <p>Попробуйте объяснить, почему происходит уменьшение времени выполнения операции (не обязательно способа замещения, остальных тоже) при добавлении строки A(1) = A(1)+sin(0).</p>

```
>> tic; L = find(A<=0.5); l = length(L);
A(L(1:l))=[]; toc
```

Elapsed time is 1.486722 seconds.

% аналогичная задача обнуления...

```
>> a = rand([1000 10000]); b = a;
>> tic; a(a<0.5) = 0; toc
```

Elapsed time is 0.660367 seconds.

```
>> a = b;
>> tic; a = a.*(a>=0.5); toc
```

Elapsed time is 0.536720 seconds.

```
>> a = b;
>> tic; a(find(a<0.5)) = 0; toc
```

Elapsed time is 1.055111 seconds.

```
function f = Fib1(n)
F = zeros(1, n+1);
F(2) = 1;
for i = 3:n+1
    F(i) = F(i-1) + F(i-2);
end
f = F(n);
%-----
function f = Fib2(n)
if n==1
    f = 0;
elseif n==2
    f = 1;
else
    f1 = 0; f2 = 1;
    for i = 2:n-1
        f = f1 + f2;
        f1 = f2; f2 = f;
    end
end
%-----
function f = Fib3(n)
if n==1
    f = 0;
elseif n==2
    f = 1;
else
    f = Fib3(n-1) + Fib3(n-2);
end
%-----
function f = Fib4(n)
A = [0 1; 1 1];
y = A^n*[1; 0];
```

Решим следующую задачу:
занулить все элементы
массива, которые меньше 0.5.

Здесь выгоднее использовать
логический массив для
обычного умножения, а не
индексации массива!

С функцией **find** опять
«ужасно»!

Это пять реализаций
вычисления **n**-го числа
Фибоначчи из [Griffiths],
[Loren]. Рекурсивная версия
(Fib3) самая медленная.
Время работы функции Fib3
при небольших **n=20** на
несколько порядков превышает
время работы остальных
функций.

Сравните время работы всех
функций при больших
значениях **n**.

Удивительно, но лучшие из
этих функций работают
быстрее, чем вычисления
 $\alpha_1 = (1 + \sqrt{5}) / 2$;
 $\alpha_2 = (1 - \sqrt{5}) / 2$;
 $c_1 = (1 - \alpha_2) / (\alpha_1 - \alpha_2)$;
 $c_2 = (\alpha_1 - 1) / (\alpha_1 - \alpha_2)$;
 $f_n = \text{round}(c_1 * \alpha_1^{(n-2)} + c_2 * \alpha_2^{(n-2)})$.
Заметим, что мы считали
первое число Фибоначчи
равное нулю, т.е. ряд этих
чисел:

0 1 1 2 3 5 8 ...

<pre>f = y(1); %----- function f = Fib5(n) F = [0 1 zeros(1,n-2)]; a = [1 -1 -1]; b = 1; F = filter(b, a, F); f = F(n);</pre>	
<pre>>> A = rand([10000 1000]); >> b = rand([10000 1]); >> tic; x = pinv(A)*b; toc Elapsed time is 37.103009 seconds. >> tic; x = inv(A'*A)*A'*b; toc Elapsed time is 6.254909 seconds. >> tic; x = inv(A'*A)*(A'*b); toc Elapsed time is 2.608321 seconds. >> tic; x = (A'*A)\A'*b; toc Elapsed time is 6.415567 seconds. >> tic; x = (A'*A)\(A'*b); toc Elapsed time is 2.265513 seconds.</pre>	<p>Пять способов решения уравнения $A \cdot x = b$ с неквадратной матрицей A. Очень часто используется в задачах линейной регрессии!</p> <p>Самый быстрый способ... Всё решают скобки!</p>
<pre>>> n = 70; e = ones(n, 1); >> T = spdiags([e,-2*e,e], [-1,0,1], n, n); >> A = full(T); b = ones(n, 1); >> s = sparse(b); >> tic, T\s; sparsetime=toc, tic, A\b; fulltime = toc sparsetime = 6.3137e-005 fulltime = 0.0769</pre>	<p>Сравнение скоростей вычислений с разреженной и обычной (полной) матрицей из [Sigmon].</p> <p>Всегда используйте специфику данных!</p>
<pre>>> a = 10; % какие-то вычисления >> a = 'A';</pre>	<p>Не меняйте тип переменной! Лучше создать новую переменную другого типа.</p>
	<p>Функции load и save быстрее функций fread и fwrite.</p>

Упомянем ещё об одном интересном случае, когда лучше не использовать встроенные «эффективные» средства системы. В задачах анализа данные часто необходимо из матрицы **x** выделять подматрицы, соответствующие одинаковым значениям элементов первого столбца (например, в первом столбце записан номер

класса, и надо быстро перечислять объекты из заданного класса). В этом случае конструкции типа

```
>> X(X(:,1)==className,:)
```

могут оказаться очень неэффективными. Гораздо лучше упорядочить матрицу (один раз!) по значениям элементов первого столбца (часто матрица уже упорядочена) и создать векторы начал и концов классов:

```
>> X = sortrows(X, 1); % сортировка строк
```

```
>> [classes indxF] = unique(X(:,1), 'first'); % классы и их начала
```

```
>> [classes indxL] = unique(X(:,1), 'last'); % последние эл-ты классов
```

Теперь доступ к классу `className` может быть реализован командами

```
>> ic = find(classes==className, 1, 'first'); % номер нашего класса
```

```
>> X(indxF(ic):indxL(ic),:) % вывод объектов
```

§21. О точности и памяти...

<pre>>> ((1e-15 + 1e-15) + 1e30) - 1e30 ans = 0 >> (1e-15 + 1e-15) + (1e30 - 1e30) ans = 2.0000e-015 >> realmax + 1000000 - realmax ans = 0 >> realmax*1.0000000000001 ans = Inf</pre>	<p>Специфика арифметики с плавающей запятой!</p> <p>Операции не ассоциативны.</p>
<pre>>> X = [1.1:0.01:10]; >> find(S==1.13) ans = Empty matrix: 1-by-0 >> X(4) ans = 1.1300 >> X(1:5)-[1.1 1.11 1.12 1.13 1.14] ans = 1.0e-015 * 0 0 0 0.2220 0.2220</pre>	<p>Пример «неверной» работы функции. Элемент 1.13 есть в массиве, но функция find его не видит. Его видит функция find(abs(S-1.13)<eps), но её использование также не всегда корректно, поскольку</p> <pre>>> 0==(0+eps/1000000000000000) ans = 0</pre> <p>Попробуйте разобраться, в чём тут проблема. См. help eps.</p>
<pre>>> ismember(0:10,10.*(0:0.1:5)) ans =</pre>	<p>Ещё одна иллюстрация этой проблемы. Сравните с действием ismember(0:10,</p>

<pre> 1 1 1 0 1 1 0 >> 3 == ((0.1*3)*10) ans = 0 >> (0.3 - 0.1*3) ans = -5.5511e-017 </pre>	<pre>round(10.*(0:0.1:5))).</pre> <p>Заметьте, что 0.3 не равно 0.1*3! Дело в том, что число 0.3 не представимо точно «в бинарной форме»...</p>
<pre> >> a = single(0.1) - 0.1 a = 1.4901e-009 </pre>	<p>Лишние цифры могут возникать при изменении точности.</p>
<pre>x = -1:0.01:1 + eps; plot(sin(x)./x);</pre>	<p>Чтобы график получался без разрывов можно сместить значения на эпсилон (совет Antenna Geek из [Loren]).</p>
<pre> % Первый способ хранения в структуре >> for i = 1:10 >> X1(i).a = 5; >> X1(i).b = 5; >> X1(i).c = 5; >> end; % Второй способ хранения в структуре for i = 1:10 X2.a(i) = 5; X2.b(i) = 5; X2.c(i) = 5; end; % хранение в массиве ячеек >> Y = struct2cell(X1); >> Y = squeeze(Y) % т.к. размер 3*1*10 Y = [5] [5] [5] [5] [5] [5] [5] [5] [5] [5] [5] [5] [5] [5] [5] [5] [5] [5] [5] [5] [5] [5] [5] [5] [5] [5] [5] [5] % хранение в массиве >> Z = cell2mat(Y) Z = 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 >> Z2 = int8(Z); >> Z3 = sparse(Z); >> whos </pre>	<p>Второй способ хранения данных (структура массивов) более предпочтителен с точки зрения хранения памяти.</p> <p>Самый экономный способ хранения данных – в массиве (правда, для разнотипных данных это не всегда возможно).</p> <p>Часто память удаётся экономить, если учесть специфику данных. Например, хранить целочисленные матрицы в формате int8, а матрицы с подавляющим числом нулей переводить в формат sparse (но если нулевых элементов мало, то возникает обратный эффект). Попробуйте набрать команды</p> <pre> >> A = sparse(round(rand([1 100]))); >> A2 = sparse(round(2/3*rand([1 100]))); >> A3 = sparse(round(3/5*rand([1 100]))); >> whos </pre>

<pre> Name Size Bytes Class Attributes X1 1x10 2232 struct X2 1x1 612 struct Y 3x10 2040 cell Z 3x10 240 double Z2 3x10 30 int8 Z3 3x10 404 double sparse i 1x1 8 double </pre>	
<pre> function f = myfunc(a,b,c) a = a + c; % new b(1) = b(1) + 1; % new!!! f = a(c>0) + b(c>0); </pre>	<p>В этой функции переменные a, b меняются, поэтому под них будут созданы новые области памяти, а под переменную c – нет.</p>
<pre> >> feature('memstats') Physical Memory (RAM): In Use: 1381 MB (56551000) Free: 1689 MB (69933000) Total: 3070 MB (bfe84000) Page File (Swap space): In Use: 1376 MB (56047000) Free: 4987 MB (137b0b000) Total: 6363 MB (18db52000) Virtual Memory (Address Space): In Use: 590 MB (24e38000) Free: 1457 MB (5b1a8000) Total: 2047 MB (7ffe0000) Largest Contiguous Free Blocks: 1. [at 1f960000] 1198 MB (4aee0000) 2. [at 7c41b000] 50 MB (32d5000) 3. [at 6f49d000] 39 MB (2763000) 4. [at 1c990000] 32 MB (2000000) 5. [at 71fb7000] 18 MB (1219000) 6. [at 775d3000] 13 MB (d0d000) 7. [at 1b0d0000] 8 MB (800000) 8. [at 6ec8b000] 7 MB (7c5000) 9. [at 7f7f0000] 7 MB (79f000) 10. [at 1a0d0000] 7 MB (700000) ===== 1382 MB (566a2000) </pre>	<p>Выводится информация о памяти. В том числе информация о свободных непрерывных блоках памяти.</p> <p>Функция pack переписывает содержимое памяти, используемое MatLabом, на диск, а затем загружает в один непрерывный блок.</p> <p>Помните, что на 32-битных платформах можно получить доступ только к 4Гб ОЗУ, а некоторые операционные системы (например Windows) снижают память, доступную приложениям, до 2Гб. В Vista с этим можно бороться командой</p> <p>BCDEdit /set increaseuserva 3072</p> <p>в XP SP2 – загрузкой ОС с параметром /3gb (в файле Boot.ini).</p>

Отметим также, что хотя использование циклов является дурным тоном в системе MatLab (если без них можно обойтись), часто циклы применяют из-за ограничений по памяти. Если не удаётся совершить операцию над матрицей больших размеров, поскольку необходимо создание дополнительных матриц, то матрицу делят на «куски» и операцию производят «по кускам» (при этом возникает цикл для перебора фрагментов матрицы).

§22. Примеры анимации, GUI

<pre>>> M = moviein(50); >> x = rand([1 10]); >> y = rand([1 10]); >> axis([-1 1 -1 1]) >> axis square >> grid off >> h = plot(x, y, '.'); >> set(h, 'MarkerSize' , 20); >> for i = 1:50 >> x = x + (rand([1 10])-0.5)/100; >> y = y + (rand([1 10])-0.5)/100; >> set(h , 'XData' , x , 'YData' , y) >> M(:, i) = getframe; >> end; >> movie(M)</pre>	<p>Здесь будет храниться 50 кадров.</p> <p>Копируем в i-й кадр.</p> <p>Напомним, что комбинация Ctrl+C не прерывает выполнение встроенной функции! Поэтому выполнение movie(M) не может быть приостановлено.</p>
<pre>>> [x y z] = deal(linspace(-1,1,50)); >> N = 20; >> M = moviein(N); >> n = 1; >> [X,Y,Z] = meshgrid(x,y,z); >> t = 0; >> k = 10; >> l = 10; >> N2 = 1; >> w = 0.5; >> m = sqrt((k^2 + l^2)*(N2-w^2)); >> while n < N >> rho = 1 + 0.1*Z; >> rhopr = -0.02*sin(k*X+l*Y+m*Z-w*t); >> Hs = slice(X,Y,Z,rho+rhopr, 0, 0, 0); >> set(Hs, 'facecolor', 'interp') >> title('Internal waves', ... 'FontName', 'Times', 'FontSize', [18]); >> box on >> axis off >> drawnow >> t = t + 0.5; >> A = getframe(1); % сохранение >> M(n) = A; >> n = n + 1; >> end</pre>	<p>Ещё пример анимации.</p>
<pre>>> b = uicontrol('Style' , 'pushbutton', 'Units' , 'normalized', 'Position', [.5 .5 .2 .1], 'String', 'Нажми меня'); >> s = 'set(b, ''Position'', [.8*rand .9*rand .2 .1])'; >> eval(s);</pre>	<p>Создание кнопки.</p> <p>Перемещение кнопки в случайные места.</p>

<pre>>> set(b, 'Callback', s); >> s = 'set(b, 'FontSize' , get(b, 'FontSize') + 1)'; >> set(b, 'Callback' , s)</pre>	<p>Обработчик события «нажатие кнопки».</p> <p>Попробуйте вызов get(b) для перечня всех свойств кнопки. Поменяйте значения некоторых свойств.</p>
<pre>>> h = waitbar(0, 'Converting...'); >> waitbar(55/100, h, 'Converting...') >> close(h)</pre>	<p>Вывод «счётчика состояния процесса» (выполняйте эти команды последовательно).</p>
<pre>>> pause</pre>	<p>Пауза в вычислениях, прерывается нажатием клавиши (полезна в m-файлах).</p>
<pre>>> xpsound</pre>	<p>Демонстрация работы со звуком.</p> <p>Основные функции: sound и wavread.</p>
<pre>>> mcc -m simple</pre>	<p>Получение exe-файла (для его работы нужен Matlab Compiler).</p>

§23. Перестановки

<pre>>> randperm(5) ans = 3 2 5 1 4</pre>	<p>Случайная перестановка. Выбор k из n элементов без повторений: X = randperm(n); x = X(1:k).</p>
<pre>>> S = 'abcd'; k = 2; >> nchoosek(S, k) ans = ab ac ad bc bd cd >> nchoosek(length(S), k) ans = 6</pre>	<p>Всевозможные сочетания по k элементов.</p> <p>Число таких сочетаний. Если n и k векторы, то можно использовать запись round(exp(gammain(n+1) - gammain(k+1) - gammain(n-k+1))).</p>
<pre>>> perms('ABC') ans = CBA CAB BCA BAC ABC ACB</pre>	<p>Всевозможные перестановки.</p>

<pre>>> factorial(3) ans = 6</pre>	Их число.
--	-----------

§24. Символьные вычисления

<pre>>> syms a b; >> simplify((a+1)*(a-1)+b*b+2*b+2) ans = a^2+1+b^2+2*b >> simplify(sin(2*a)/cos(a)) ans = 2*sin(a) >> expand((a+1)*(a-1)*(b+2)) ans = a^2*b+2*a^2-b-2 >> factor(a^3-b^3) ans = (a-b)*(a^2+a*b+b^2) >> subs(sin(a)+sin(b), b, pi) ans = sin(a) >> diff(sin(a^3)) ans = 3*cos(a^3)*a^2 >> int(sin(a)^2) ans = -1/2*sin(a)*cos(a)+1/2*a >> q = solve('a+b=1', 'a^2-b=2'); >> q.a ans = -1/2-1/2*13^(1/2) -1/2+1/2*13^(1/2) >> q.b ans = 3/2+1/2*13^(1/2)</pre>	<p>Пример символьных вычислений.</p> <p>Упрощение выражений.</p> <p>Раскрытие выражений (ср. с упрощением).</p> <p>Разложение на множители.</p> <p>Подстановка.</p> <p>Производная.</p> <p>Интеграл.</p> <p>Системы уравнений.</p>
--	--

<pre> 3/2-1/2*13^(1/2) >> dsolve('Da+a=exp(t)') ans = 1/2*exp(t)+exp(-t)*C1 </pre>	<p>Дифференциальные уравнения.</p>
<pre> >> syms x1 x2 x3 >> W = [1 x1 x1^2; 1 x2 x2^2; 1 x3 x3^2] W = [1, x1, x1^2] [1, x2, x2^2] [1, x3, x3^2] >> pretty(W) [2] [1 x1 x1] [] [2] [1 x2 x2] [] [2] [1 x3 x3] >> a = det(W) a = x2*x3^2-x2^2*x3- x1*x3^2+x1^2*x3+x1*x2^2-x1^2*x2 >> factor(a) ans = -(-x2+x1)*(x3-x2)*(x3-x1) </pre>	<p>Пример символьного вычисления определителя Вандермонда.</p> <p>«Удобный» вид арифметических выражений.</p>
<pre> >> syms x; >> limit([sin(x)/x, (1+x)^(1./x)]) ans = [1, exp(1)] >> Taylor(sin(x), x, 0, 8) ans = x-1/6*x^3+1/120*x^5-1/5040*x^7 >> syms x y z; >> laplace(x+y*z) ans = 1/s^2+y*z/s </pre>	<p>Вычисление пределов.</p> <p>Разложение в ряд Тейлора в окрестности нуля (выписать 8 первых членов разложения).</p> <p>Преобразование Лапласа.</p>
<pre> >> x = linspace(0,pi,100); >> trapz(x,sin(x)) </pre>	<p>«Несимвольно» интегралы берутся так...</p> <p>Интегрирование методом</p>

<pre>ans = 1.9998 >> quad('sin(x)',0,pi) ans = 2.0000</pre>	<p>трапеций.</p> <p>Интегрирование по формуле Симпсона (с автоматическим выбором шага).</p>
---	---

§25. Задания для самостоятельной работы

Для начала разберём подробно решение следующей задачи [CVD] (попробуйте её решить, не заглядывая в ответ): заполнить в векторе **x** все нулевые значения предыдущими ненулевыми значениями. Для вектора **x** = [7 0 0 1 0 0 0 3 0] должен получиться ответ **x** = [7 7 7 1 1 1 1 3 3]. Решение состоит из трёх команд:

<pre>>> x = [7 0 0 1 0 0 0 3 0]; >> valind = find(x) valind = 1 4 8 >> x(valind(2:end)) = diff(x(valind)) x = 7 0 0 -6 0 0 0 2 0 >> x = cumsum(x) x = 7 7 7 1 1 1 1 3 3</pre>	<p>Находим ненулевые позиции.</p> <p>Подготавливаем вектор для действия кумулятивной суммы... заменяем элемент разностью между ним и предыдущим ненулевым.</p> <p>Теперь вычисляем кумулятивную сумму и получаем ответ.</p>
--	---

Здесь «ключевой» является вторая команда. Чтобы решить задачу, необходимо догадаться, что следует применять функцию **cumsum**. Это не так сложно сделать, поскольку, когда мы её применяем к вектору, соседние нулевые позиции она заполняет одинаковыми значениями. Осталось добиться, чтобы эти значения были «нужными».

При решении приведённых ниже задач нельзя пользоваться циклами и условными операторами. Прочитайте условие в правой колонке, попробуйте решить самостоятельно. Попробуйте не просто решить, а дать компактное (минимальное число строк кода) и эффективное (быстро выполняется даже при больших размерах входных данных) решение. Ответ приводится в левой колонке.

Ответ	Задача
<pre>>> X = setdiff(2:2:N, 6:6:N) % гораздо хуже: >> setdiff(2*(1:fix(N/2)), 3*(1:fix(N/3)))</pre>	<p>Выписать все чётные числа от 1 до N, которые не делятся на 3.</p>
<pre>>> ismember(A, B1) ismember(A, B2)</pre>	<p>Пометить в матрице A все элементы, перечисленные в</p>

	<p>векторах B1 и B2. Для B1 = $\begin{bmatrix} 1 & 2 \\ 2 & 3 & 4 \\ 1 & 2 & 2 \\ 3 & 3 & 5 \end{bmatrix}$</p> <p>ответ</p> $\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}$
<p>% в одну строчку:</p> <pre>>> reshape(repmat(X,N,1), 1, N*length(X));</pre> <p>% через вспомогательную переменную и индексный вызов:</p> <pre>>> A = X(ones(1,N),:); >> A = A(:).'</pre>	<p>В вектор строке X повторить все значения N раз. При N = 3, X = [2 0 1] ответ</p> <p>ans =</p> $\begin{bmatrix} 2 & 2 & 2 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}$
<pre>>> dec2bin(0:(2^n-1))-'0'</pre> <p>% чуть сложнее...</p> <pre>>> +(dec2bin(0:(2^n-1))=='1')</pre> <p>ans =</p> $\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$	<p>Построить матрицу (типа <i>double</i>), в которой по строкам записаны все n-мерные бинарные векторы.</p>
<pre>>> X'*Y</pre> <p>% типичная ошибка – использование repmat!</p>	<p>Построить таблицу умножения всевозможных пар элементов таких, что первый берётся из множества X, а второй – из Y. Например, при X = [-1 0 1], Y = [2 3 5]</p> <p>ответ</p> <p>ans =</p> $\begin{bmatrix} -2 & -3 & -5 \\ 0 & 0 & 0 \\ 2 & 3 & 5 \end{bmatrix}$
<pre>>> cumsum(ones(n))</pre> <p>% менее эффективный вариант:</p> <pre>>> tril(ones(n))*ones(n)</pre>	<p>Для натурального числа n построить матрицу размера n×n, все элементы i-й строки которой равны i. Например, при n = 4</p> <p>ans =</p> $\begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 \\ 4 & 4 & 4 & 4 \end{bmatrix}$ <p>Команда порождения должна «умещаться» в одной строке.</p>

	Можно использовать любые стандартные функции, кроме for , repmat и meshgrid .
<pre>>> b = A==max(A); >> cumsum(b).*b</pre>	<p>Занумеровать в порядке следования все максимальные элементы в векторе A. Например, для вектора A = [1 2 3 3 2 1 3 1] получить [0 0 1 2 0 0 3 0].</p>
<pre>>> X = triu(repmat(1:n, n, 1)); >> X(X(:)~=0)</pre>	<p>Для натурального числа n породить вектор, в котором сначала записана единица, потом две двойки, потом три тройки и т.д. Последние элементы вектора – n n-ок. Например, при n = 4</p> <pre>X = 1 2 2 3 3 3 4 4 4 4</pre>
<pre>>> [X Y] = meshgrid(A, B); >> max(max(sin(X + Y)))</pre>	<p>Дано множество A и множество B. Найти максимум функции sin(a+b) при a из A и b из B.</p>
<pre>>> max(max(A)-min(B), max(B)-min(A)) % функция abs не используется!</pre>	<p>Дано множество A и множество B. Найти (самым эффективным способом) максимум функции abs(a-b) при a из A и b из B.</p>
<pre>>> max(A)<=min(B)</pre>	<p>Самым простым способом выяснить, верно ли, что все элементы множества B не меньше всех элементов множества A.</p>
<pre>>> [X Y] = meshgrid(1:n, 1:m); >> abs(X - Y)</pre>	<p>Придумать универсальный способ реализации матриц, в которых ij-й элемент равен расстоянию до главной диагонали в метрике L1 (без использования функции toeplitz). Пример матрицы:</p> <pre>ans = 0 1 2 3 4 1 0 1 2 3 2 1 0 1 2</pre>
<pre>>> [x y] = meshgrid(A, B); >> X = [x(:), y(:)]</pre>	<p>Для двух множеств A, B сформировать матрицу, в которой по строкам записаны все пары – элементы декартова произведения этих множеств. Например, для множеств A = [2 3], B = [1 2] ответ</p> <pre>X =</pre>

	<pre> 2 1 2 2 3 1 3 2 </pre>
<code>>> max(A(isprime(A)))</code>	Найти максимальный простой элемент множества чисел A .
<pre> >> [a a a] = unique(a) % обходной путь... >> [b i] = sort(a); >> b = cumsum([1 diff(b)]>0); >> a(i) = b </pre>	Заменить элементы вектор-строки a на индексы соответствующих элементов вектора unique(a) . Для строки a = [9 2 5 6 2 5 5] ответ – <pre> a = 4 1 2 3 1 2 2 </pre>
<pre> >> [B I] = unique(A, 'first'); >> [B J] = unique(A, 'last'); >> B(I==J) % другой вариант >> b=sort(A); >> setdiff(b, b([1 diff(b)]==0))) </pre>	Во множестве A оставить только уникальные элементы (входят только один раз). Ответ для множества A = [3 2 1 3 4 4 7 5 5] : <pre> ans = 1 2 7 </pre>
<code>>> sum(A=='a')</code>	Для строки A (например для A = 'abbcaaccab') посчитать число вхождений буквы 'a' (самым простым и быстрым способом).
<pre> >> n = 2; >> b = (2.^(0:n))'; >> A = repmat((0:n)', 1, n+1); >> A = A.^(A'); >> A\b ans = 1.0000 0.5000 0.5000 </pre>	Дано натуральное число n . Найти коэффициенты полинома степени n , который в точках 0,1,...,n принимает значения 1,2,4,8,...,2^n (все степени двойки). Написать ответ при n = 2 .
<code>>> all(A == A(end:-1:1))</code>	Проверить, является ли вектор A симметричным. Например, векторы A = [3 4 5 4 3] , A = [6 6] , A = [7] являются, а векторы A = [1 2] , A = [1 2 3 4 1] – нет.
<pre> >> sort([A, B]) % без sort не будет отсортированным </pre>	Как объединить два множества с учётом кратности элементов, т.е. объединение A = [2,1,3,4,3] и B = [2,3,5] даст [1, 2, 2, 3, 3, 3, 4, 5] ?
<code>>> sum(B(:) == 2) == 2</code>	Как проще всего проверить условие «в матрице B только два элемента равны двум»?
<pre> >> X = repmat(1:n, n, 1); >> min(X, X') </pre>	Придумайте способ порождения матриц вида

	<pre>ans = 1 1 1 1 2 2 1 2 3 ans = 1 1 1 1 1 2 2 2 1 2 3 3 1 2 3 4</pre>
<pre>S(1, :) = s1; S(2, 1:length(s2)) = s2; S = S(:)'; S(S==0) = []</pre>	<p>Написать код, который соединяет две строки s1, s2 следующим образом: [s1(1) s2(1) s1(2) s2(2) ...]. Для s1 = '12345', s2 = 'abc' должна получаться строка '1a2b3c45', а для s1 = '123', s2 = 'abcde' – строка '1a2b3cde'.</p>
<pre>>> [sortS i] = sort(double(S)); >> S(i([1, diff(sortS)] == 0)) = [] % без приведения double не получится сделать diff</pre>	<p>Оставить в строке S только первые вхождения всех элементов. Для S = 'try to find unique elements' ответ S = try ofindugelms (ответ не отсортирован в общем случае).</p>
<pre>% Решение из [Loren] >> i = any([a;b] == 0) >> a(i) = [] >> b(i) = [] % Другой вариант решения, который можно % улучшить, имея априорные сведения % о количестве нулей в векторах >> i = find(a~=0); >> i = i(b(i)~=0); >> a = a(i); >> b = b(i);</pre>	<p>Оставить в двух векторах a, b, содержащих одинаковое число элементов, только те элементы, которые соответствуют позициям ненулевых элементов в обоих векторах. Для a = [NaN 1 2 0 0 Inf 0]; b = [1 0 3 4 0 0 NaN]; ответ a = NaN 2 b = 1 3</p>
<pre>>> uS = unique(S)'; >> [m i] = max(sum((repmat(uS, 1, size(S,2)) == repmat(S, length(uS), 1)))); >> [uS(i) m] % подумайте, нельзя ли улучшить этот метод</pre>	<p>Для мультимножества S определить, какой элемент встречается в нём наибольшее число раз. Вывести этот элемент и его кратность. Для S = [1 1 3 2 3 3 1 3 9] ответ – ans = 3 4 (элемент 3 встретился 4 раза). Для S = [2 1 2 1] ответ – [1 2] или [2 2].</p>

<pre>>> isequal(sort(A), sort(B)) % обходной путь >> (length(A) == length(B)) && all(sort(A) == sort(B))</pre>	<p>Сравнить, совпадают ли два мультимножества? Например, мультимножества A = [1 2 3 1 1 2] и B = [3 1 1 1 2 2] равны, а A = [1 2 3 1 2] и B = [1 1 3 2 3] – нет.</p>
<pre>>> max(A(find(A(1:end-1)==0)+1))</pre>	<p>Найти максимальный элемент в вектор-строке среди элементов, перед которым стоит нулевой. Для A = [6,2,0,3,0,0,5,7,0] ответ ans = 5</p>
<pre>% этот способ не рекомендуют из-за лишних умножений... >> X = val.*ones(siz); % аналог со сложениями >> X = val + zeros(siz); % часто рекомендуемый способ >> X(prod(siz)) = val; >> X = reshape(X, siz); >> X(:) = X(end); % улучшение предыдущего варианта >> X(1:prod(siz)) = val; >> X = reshape(X,siz) % один из самых быстрых способов >> X = zeros(siz); X(:) = val; % самый «естественный» способ >> X = repmat(val,siz); % самый плохой (долгий) способ >> X = val(ones(siz));</pre>	<p>Реализуйте различные способы генерации массива размера siz с элементами, равными val. Чем больше, тем лучше. При siz = [2 3] и val = 5 ответ X = 5 5 5 5 5 5</p>
<pre>>> X(mod((1:end)-k-1, end)+1)</pre>	<p>Реализуйте в одну строчку сдвиг вектор-строки X на k. Например, при X = 'MatLab' и k = 2 результат - 'abMatL', а при k = -1 - 'atLabM'. Нельзя использовать функции MatLaba, содержащие в названии слово shift.</p>
<pre>>> X = ceil(s*rand([m n]));</pre>	<p>Сгенерируйте случайную матрицу размера m×n с элементами из 1:s.</p>
<pre>>> A(1: (size(A,1)+1): (size(A,1)*min(size(A))))</pre>	<p>Реализуйте функцию diag(A) через другие функции MatLaba.</p>
<pre>>> Y = reshape(X, [size(X,1) size(X,2)* size(X,3)])</pre>	<p>Заданы двумерные матрицы A, B, C, D одинаковых размеров. По матрице X = cat(3, A, B, C, D) получите матрицу [A, B, C, D].</p>
<pre>>> Y = permute(X,[1 3 2]); >> Y = reshape(Y, [size(X,1)* size(X,3) size(X,2)])</pre>	<p>Заданы двумерные матрицы A, B, C, D одинаковых размеров. По матрице X = cat(3, A, B</p>

	<i>С, D) получите матрицу [A; B; C; D]</i>																																				
<pre>>> B = repmat(A, [m n])</pre>	Реализуйте присваивание B = kron(ones(m,n), A) более простым способом для любых натуральных m, n и матрицы A . Например, при m = 2; n = 3; A = [4 5]; B = <table><tr><td>4</td><td>5</td><td>4</td><td>5</td><td>4</td><td>5</td></tr><tr><td>4</td><td>5</td><td>4</td><td>5</td><td>4</td><td>5</td></tr></table>	4	5	4	5	4	5	4	5	4	5	4	5																								
4	5	4	5	4	5																																
4	5	4	5	4	5																																
<pre>>> B = A(repmat(1:size(A,1),m,1), repmat(1:size(A,2),n,1)) % Менее эффективный способ из [Acklam] >> i = 1:size(A,1); >> j = 1:size(A,2); >> B = A(i(ones(1,m)), j(ones(1,n)))</pre>	Реализуйте присваивание B = kron(A, ones(m,n)) более простым способом для любых натуральных m, n и матрицы A . Например, при m = 2; n = 3; A = [4 5; 6 7]; B = <table><tr><td>4</td><td>4</td><td>4</td><td>5</td><td>5</td><td>5</td></tr><tr><td>4</td><td>4</td><td>4</td><td>5</td><td>5</td><td>5</td></tr><tr><td>6</td><td>6</td><td>6</td><td>7</td><td>7</td><td>7</td></tr><tr><td>6</td><td>6</td><td>6</td><td>7</td><td>7</td><td>7</td></tr></table>	4	4	4	5	5	5	4	4	4	5	5	5	6	6	6	7	7	7	6	6	6	7	7	7												
4	4	4	5	5	5																																
4	4	4	5	5	5																																
6	6	6	7	7	7																																
6	6	6	7	7	7																																
<pre>>> B = reshape([repmat([A; repmat(zeros(size(A)), n, 1)], n-1, 1); A], n*size(A,1), n*size(A,2)); >> I = reshape((1:n*size(A,2))', n, size(A,2))'; >> B = B(:,I); % Более эффективный и эффективный способ из [Acklam] >> B = zeros([size(A) n n]); >> B(:,:,1:n+1:n^2) = repmat(A, [1 1 n]); >> B = permute(B, [1 3 2 4]); >> B = reshape(B, n*size(A)); % Самый эффективный и эффективный способ % (формирование «нужной» команды) >> eval(['B = blkdiag(' repmat('A, ' ,1,n-1) 'A);'])</pre>	Реализуйте присваивание B = kron(eye(n),A) более простым способом для любых натуральных m, n и матрицы A . Например, при n = 3; A = [4 5; 6 7]; B = <table><tr><td>4</td><td>5</td><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>6</td><td>7</td><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>4</td><td>5</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>6</td><td>7</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>4</td><td>5</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>6</td><td>7</td></tr></table>	4	5	0	0	0	0	6	7	0	0	0	0	0	0	4	5	0	0	0	0	6	7	0	0	0	0	0	0	4	5	0	0	0	0	6	7
4	5	0	0	0	0																																
6	7	0	0	0	0																																
0	0	4	5	0	0																																
0	0	6	7	0	0																																
0	0	0	0	4	5																																
0	0	0	0	6	7																																
<pre>% Способ из [Acklam] >> B = zeros([size(A) n n]); >> B(:,:,1:n+1:n^2) = repmat(A, [1 1 n]); >> B = permute(B, [3 1 4 2]); >> B = reshape(B, n*size(A));</pre>	Реализуйте присваивание B = kron(A,eye(n)) более простым способом для любых натуральных m, n и матрицы A . Например, при n = 3; A = [4 5; 6 7]; B = <table><tr><td>4</td><td>0</td><td>0</td><td>5</td><td>0</td><td>0</td></tr><tr><td>0</td><td>4</td><td>0</td><td>0</td><td>5</td><td>0</td></tr><tr><td>0</td><td>0</td><td>4</td><td>0</td><td>0</td><td>5</td></tr><tr><td>6</td><td>0</td><td>0</td><td>7</td><td>0</td><td>0</td></tr><tr><td>0</td><td>6</td><td>0</td><td>0</td><td>7</td><td>0</td></tr><tr><td>0</td><td>0</td><td>6</td><td>0</td><td>0</td><td>7</td></tr></table>	4	0	0	5	0	0	0	4	0	0	5	0	0	0	4	0	0	5	6	0	0	7	0	0	0	6	0	0	7	0	0	0	6	0	0	7
4	0	0	5	0	0																																
0	4	0	0	5	0																																
0	0	4	0	0	5																																
6	0	0	7	0	0																																
0	6	0	0	7	0																																
0	0	6	0	0	7																																

<pre>>> reshape(permute(repmat(A, [1 1 size(B, 1) size(B, 2)]), [3 1 4 2]),size(A, 1)*size(B, 1), size(A, 2)* size(B, 2)).* repmat(B, [size(A, 1) size(A, 2)])</pre>	<p>Реализуйте функцию kron(A, B) для любых двумерных матриц через функции reshape, permute, repmat, size. Для A = [2 3; 4 5], B = [1 7; 6 0] должна получаться матрица</p> <pre>ans = 2 14 3 21 12 0 18 0 4 28 5 35 24 0 30 0</pre>
<pre>>> D = C(:,size(C,2):-1:1)'</pre>	<p>Реализуйте присваивание D = rot90(C), не используя функции rot90.</p>
<pre>>> D = C(size(C,1):-1:1,size(C,2):-1:1)</pre>	<p>Реализуйте поворот матрицы на 180 градусов без использования функции rot90. При C = [1 2 3; 4 5 6] результат поворота</p> <pre>D = 6 5 4 3 2 1</pre>
<pre>>> all(x(:) > 0) % или так (хуже) >> sum(x(:)<=0)==0</pre>	<p>Напишите код, который проверяет, что в матрице x все элементы положительные.</p>
<pre>>> prod(size(x))<=1 % обратите внимание, что size([])= [0 0]</pre>	<p>Напишите код, который проверяет, что матрица x является скаляром или пустой матрицей.</p>
<pre>>> ndims(x) <= 2 & sum(size(x) > 1) <= 1</pre>	<p>Напишите код, который проверяет, что x является вектор-столбцом или вектор-строкой (быть может, пустой).</p>
<pre>>> m = 10; n = 2; >> X = rand([m n]); >> y = rand([1 n]); >> [m i] = min(sum((X - repmat(y, m, 1)).^2, 2)); >> scatter(X(:,1),X(:,2)); >> hold on; >> scatter(y(:,1),y(:,2),'filled'); >> line ([X(i,1) y(1)],[X(i,2) y(2)])</pre>	<p>Сгенерируйте на плоскости m случайных точек. Для новой случайной точки найдите среди этих точек ближайшую (реализуйте метод ближайшего соседа) по евклидовой метрике. Сделайте визуализацию.</p>
<pre>>> D = reshape(sqrt(sum((repmat(X, size(Y,1), 1) - repmat(Y', size(X,1), 1), size(X, 1)*size(Y, 1), size(X, 2))).^2, 2)), size(X, 1),size(Y, 1)) % Эффектный способ из [Acklam]</pre>	<p>Для матрицы X = rand([m n]) и матрицы Y = rand([h n]), в которых по строкам перечислены координаты n-мерных точек постройте m*h-матрицу попарных евклидовых расстояний.</p>

<pre>>> D = sqrt(sum(abs(repmat(permute(X, [1 3 2]), [1 size(Y,1) 1]) - repmat(permute(Y, [3 1 2]), [size(X,1) 1 1])).^2, 3))</pre>	
<pre>>> [x(:,2) x(:,1)] = find(tril(ones(n), -1))</pre>	<p>Сгенерировать все сочетания из n элементов по два, не используя специальных функций для работы с перестановками (т.е. реализовать функцию nchoosek(1:n, 2)).</p>
<pre>% Первый способ >> m = length(a); >> len = b - a + 1; >> n = sum(len); >> c = ones(1, n); >> c(1) = a(1); >> len(1) = len(1) + 1; >> c(cumsum(len(1:end-1))) = a(2:m) - b(1:m-1); >> c = cumsum(c); % Эффективное решение методом формирования «нужной» команды >> c = eval(['[', sprintf('%d:%d ',[a;b]), ', ']]); % Ещё одно эффективное решение из [Loren], предложенное Matt Fig и Jason Merrill. >> f = @(a,b) cell2mat(arrayfun(@(x,y) {x:y}, a, b)); >> c = f(a,b);</pre>	<p>Для двух векторов одинаковой длины a и b сгенерировать вектор c = [a(1):b(1) a(2):b(2) ...]. Для a = [1 3 5 6], b = [3 3 5 7] ответ</p> <pre>c = 1 2 3 3 5 6 7</pre>
<pre>>> A = repmat((1:max(a))', 1, length(a)); >> A(A>repmat(a, max(a),1)) = 0</pre>	<p>Для вектор-строки a с натуральными элементами сгенерировать матрицу, у которой в первом столбце записаны числа от 1 до a(1), во втором – от 1 до a(2) и т.д. При этом матрица дополнена нулевыми элементами. Так, для a = [3 1 2 3] должна порождаться матрица</p> <pre>A = 1 1 1 1 2 0 2 2 3 0 0 3</pre>
<pre>function m = mymean(X) L = isnan(X); X(L) = 0; m = sum(X)./sum(~L);</pre>	<p>Напишите функцию mymean, которая находит средние значения (по одному направлению) с учётом NaN элементов матрицы. Для</p> <pre>X = NaN 1 2</pre>

	<pre>NaN 0 6 1 5 NaN</pre> <pre>>> mymean(X) ans = 1 2 4</pre>
<pre>>> H = min(max(H, minH), maxH)</pre>	<p>В матрице H заменить все значения, которые больше maxH на maxH, а все значения, которые меньше minH на minH. Например, при H = [1 5 3; 5 2 8; 7 9 10], maxH = 8, minH = 4 должна получиться матрица</p> <pre>H = 4 5 4 5 4 8 7 8 8</pre>
<pre>>> B = cumsum(A); >> diff([0 B(cumsum(a))])</pre>	<p>Для вектор-стоек A и a вычислить сумму первых a(1) элементов вектора A, сумму следующих a(2) элементов вектора A и т.д. Например, при A = 1:10, a = [2 1 3] ответ</p> <pre>ans = 3 3 15</pre>
<pre>>> i = abs(x1-y)>abs(x2-y); >> a = x1; >> a(i) = x2(i);</pre>	<p>Даны вектор-строки x1, x2 и y. Сформировать вектор-строку a, в которой a(i) = x1(i), если y(i) ближе к x1(i), а иначе a(i) = x2(i). Для x1 = [1 2 3 4 5], x2 = [3 1 2 1 2], y = [0 1 2 3 4] ответ</p> <pre>a = 1 1 2 4 5</pre>
<pre>>> A(:,~v) = 0 % чуть хуже вариант из [Cambridge] >> A = A*diag(v) % и совсем плохо: >> A = A.*repmat(v,size(A,1),1) % возможное решение >> A = bsxfun(@(x,y) x.*y, A, v);</pre>	<p>Дана матрица A и бинарная вектор-строка v. В матрице оставить без изменения все столбцы, которые соответствуют единицам вектора v, а элементы остальных столбцов занулить. Для v = [0 1 0 1] и A = [1:4;3:6] ответ</p> <pre>A = 0 2 0 4 0 4 0 6</pre>
<pre>>> i = (a == 0); >> a(i(1:end-1)&i(2:end)) = []</pre>	<p>Все нули, идущие подряд в векторе a, заменить одним нулём. Для a = [0 0 1 2 0 0 0 3 0 4 0 0] ответ</p>

	<div>a = 0 1 2 0 3 0 4 0</div>
<div>>> Z = sum(X*W.*conj(Y), 2); % менее эффективное решение >> Z = diag(X*W*Y');</div>	<div>Реализовать эффективно код [Acklam] Z = zeros(m, 1); for i = 1:m Z(i) = X(i,:)*W*Y(i,:); end</div>
<div>% вариант из [Cambridge] >> K = X*X'; >> d = diag(K); >> one = ones(length(d), 1); >> D = sqrt(d*one'+one*d'-2*K);</div>	<div>Построить матрицу попарных евклидовых расстояний системы точек, которые по координатам записаны в матрице X.</div>
<div>% решение [Acklam] >> [i, j, v] = find(X); >> t = logical(diff([0; j])); >> i = i(t)'; >> v = v(t)';</div>	<div>В каждом столбце матрицы X есть ненулевой элемент. Найти порядковые номера (в столбце) и значения всех первых ненулевых элементов каждого столбца. Для X = <div><div>020</div><div>003</div><div>445</div></div><div>ответ</div><div><div>312</div><div>423</div></div>(т.е. 3й элемент первого столбца равен 4, 1й элемент второго – 2 и т.д.)</div>
<div>>> B = bsxfun(@(x,y) x.*(x==y), A, max(A,[],2))</div>	<div>В каждой строке матрицы A оставить неизменным максимальный элемент, а остальные обнулить. Ответ записать в новую матрицу B. Для A = <div><div>8940</div><div>7832</div><div>8316</div></div><div>ответ B =</div><div><div>0900</div><div>0800</div><div>8000</div></div></div>

§26. Как не надо программировать в системе MatLab

Весь предыдущий текст направлен на обучение программированию на примерах. Описание MatLab-команд есть в хорошо организованных файлах помощи (большинство современных книг по этой системе просто «повторяют» их содержание). Язык системы очень прост в освоении. Достаточно серьезные программы удаётся начинать писать прямо сразу после знакомства с системой MatLab¹. Однако, несмотря на простоту освоения языка, есть куча тонкостей, которыми надо овладеть... Им и было уделено основное внимание. Практика

¹ Как правило, пользователи уже знают язык C.

показывает, что большинство студентов не может сразу проникнуться всеми тонкостями. Ниже приведены примеры типичных ошибок студентов, только что «освоивших» язык.

Написано	Верный вариант
>> A = reshape(A, [1 size(A,1)*size(A,2)]) % для двумерной матрицы A	>> A = A(:)'
>> b = size(x); >> b(end) % для вектор-строки x	>> b = length(x);
>> A - tril(A)	>> triu(A,1)
>> sum(a==b) == size(a, 2) % для вектор-строк a и b	>> isequal(a,b) % универсальный способ!
>> sum((a - b) ~= 0) == 0	>> all(a == b)
>> X = X(:)'; X(X==0) = []	>> X(X(:)'~=0)
>> logical(A + B) % для логических массивов A и B	>> A B
>> nz = 2:length(A); >> nz(A(1:end-1) == 0)	>> find(A(1:end-1) == 0) + 1 % часто так лучше!
>> find(ismember(A,0))	>> find(A==0) % есть ещё решение через % строковые функции
>> min(x(:)) > 0 >> all(all(x>0)) >> (length(x(:))-length(x(x>0))) == 0 % сложно поверить, что это написал человек хоть что-то понимающий в программировании	>> all(x(:) > 0)
>> max(a(find(a==0) + 1)) % для вектор-строки a	Возможен выход за пределы массива (при a = [0]). Верно: >> max(a(find(a(1:end-1)==0)+1))
>> [x y] = meshgrid(1:n, 1:n); >> x = x - y; % y потом не используется	>> x = repmat(1:n,n,1); >> x = x - x'; ИЛИ >> x = -ones(n); >> x(1,:) = 0:n-1; >> x = cumsum(x);
>> cumsum(A==b).*(A==b)	>> z = A==b; cumsum(z).*(z)
>> setdiff(A, intersect(A,B))	>> setdiff(A, B)
>> [B I] = unique(A) >> A(I(I>=5))	>> [B I] = unique(A) >> B(I>=5)

§27. Советы по оформлению m-файлов

1. Имена переменных и функций должны быть «говорящими» (примеры имён переменных: `fileName`, `maxValue`, примеры имён функций: `classify`, `selectFeatures` и т.д.).

2. При именовании переменных и функций придерживайтесь определённого стиля (который принят в Вашей группе разработчиков). Опишем некоторые наиболее часто встречающиеся рекомендации... Переменные, которые в цикле меняют свои значения, лучше начинать с букв `i`, `j`, `k`; переменные, обозначающие число, – с букв `n`, `m`. Если функция возвращает только указатель или ничего не возвращает, то её название отражает, что она делает (`plot`). Если функция возвращает одно значение, то её название отражает смысл этого значения (`max`, `min`, `mean`). В названиях не используют отрицаний (`isFound`, но не `isNotFound`). Часто используют префиксы:

префикс (примеры)	используют
<code>get</code> , <code>set</code> <code>(getobj(), setappdata())</code>	при доступе к объекту или свойству;
<code>compute</code> <code>(computeweights(), computespread())</code>	для отражения, что вычисление функции может занять достаточно много времени;
<code>find</code> <code>(findnearestneighbor(), findheaviestelement())</code>	когда что-то «ищется»;
<code>is</code> , <code>has</code> , <code>can</code> , <code>should</code> <code>(iscomplete(), hasLicense(), canEvaluate())</code>	когда возвращается логическая переменная.

Фрагмент кода

```

for iFile = 1:nFiles
    for jPosition = 1:nPositions

        fileName = getname(); % Имя функции - маленькими буквами
        fileSize = filesize(); % Не надо fsz
        if (checkTiffFormat()) % Не checkTIFFFormat!
            tableNo = % определение номера таблицы

            % ...

            for i = 1:MAX_ITERATIONS % Константа заглавными буквами
                Segment.length = % Название структуры заглавными
                                % Не надо Segment.segmentlength
            end;
        end; % if (checkTiffFormat())

    end; % for jPosition = 1:nPositions
end; % iFile = 1:nFiles

```

3. Программа должна быть разбита на «небольшие» части: функции. Если какой-то кусок кода встречается несколько раз, то неплохо бы сделать его отдельной

функцией. Если функция используется только в теле другой функции, то надо сделать её встроенной (описать в том же файле).

4. При передаче данных функции лучше использовать аргументы, а не глобальные переменные. Использование структур помогает избавиться от длинных списков аргументов.

5. Используйте уже существующие функции, если их можно эффективно применить для решения Вашей задачи. Много известных алгоритмов уже реализовано (если не в самой системе MatLab, то в одной из её библиотек). Для поиска нужных функций научитесь пользоваться помощью системы.

6. Не ленитесь документировать функцию! Не оставляйте это «на потом». Назначения всех переменных должны быть описаны. Заголовок функции должен поддерживать использование **help** (выводит первый непрерывный блок комментариев) и **lookfor** (сканирует первую строку файла). Кстати, в поле **Description** окна **Current Directory** выводится первый комментарий, найденный в тексте файла (это очень удобно при просмотре содержимого каталога).

Схема для заголовка функции

```
%FINDCLASS Классификация выборки методом MVS.
% function [classes, StructInfo] =
% findClass(trainData, trainMarks, testData)
%
% DESCRIPTION:
% Описание функции
%
% INPUTS:
% Описание входных данных. Вот пример...
% trainData - ОБУЧЕНИЕ - двумерная матрица "объект-признак".
% trainMarks - МЕТКИ ОБУЧЕНИЯ - вектор-столбец.
% testData - КОНТРОЛЬ - двумерная матрица "объект-признак".
% Обе матрицы имеют одинаковое число столбцов.
% Возможны пропуски значений (NaN).
% Вектор trainMarks имеет длину равную числу строк trainData.
%
% OUTPUTS:
% Описание выходных данных.
%
% USAGE:
% Примеры использования функции.
%
% NOTES:
% Особенности функции.
%
% Needs:
% Перечень файлов, необходимых для работы функции.
%
% See also:
% Перечень «похожих» функций.
```

```
%
% GLOBALS:
% Перечень и описание глобальных переменных.
%
% HISTORY:
% История изменений функции.
%
% AUTHOR:
% Данные автора (+ почта для связи).
%
% REFERENCES:
% Необходимые ссылки
function [classes, StructInfo] = findClass(trainData, trainMarks,
testData)
```

7. Используйте выравнивание, отступы и пробелы. Это повышает «читабельность». Отделяйте пробелом комментарий от знака %, отделяйте левую и правую части присваивания от знака = и т.д. Используйте скобки (особенно в логических выражениях). Это понятно и позволяет избежать ошибок. Избегайте слишком длинных строк.

Пример грамотного выравнивания

```
weightedPopulation = (doctorWeight * nDoctors) + ...
                     (lawyerWeight * nLawyers) + ...
                     (chiefWeight * nChiefs);
```

8. В случае ошибки желательно, чтобы функция её диагностировала и выдавала пользователю рекомендации по исправлению. Не забывайте про **try**, **catch**, **end**. Проверяйте число аргументов функции. Всегда «ожидайте ошибки» (например, условный оператор **switch** должен включать условие **otherwise**).

9. Помните, что некоторые функции ухудшают «читабельность» (**eval**, **feval**, **exist**, **break**, **continue**). Переменная, которая «создаётся» в цикле, должна быть инициализирована непосредственно перед этим циклом:

```
result = zeros(nEntries,1);
for index = 1:nEntries
    result(index) = computeFactors(index);
end
```

10. Не дублируйте данные.

```
% плохо
trainObjects = [.1 .23 .2 .15 .7];
etalons = [.1 .2 .7];

% хорошо
trainObjects = [0.1 0.23 0.2 0.15 0.7]; % и 0 надо указывать!
indexOfEtalons = [1 3 5];
```

11. Пишите «минимальный» код (по числу команд) и простой код (не используйте слишком сложные синтаксические конструкции, которые не свойственны языку).
12. Не решайте сразу слишком общую задачу – решайте свою (в интерпретаторе код, который решает общую задачу, очень сложно отладить). Сразу проверяйте код, написанный для решения какой-то подзадачи, на серии тестов.
13. Твёрдо знайте базовые команды и специфику их использования (см. весь текст выше). Например, старайтесь использовать функции, а не скрипты, структуры массивов, а не массивы структур, и т.д.

Ещё больше хороших советов можно найти в [Robbins], [Johnson].

§28. Аналоги системы MatLab

GNU Octave (<http://www.gnu.org/software/octave/>),

Scilab (<http://www.scilab.org/>),

Java-MathLib (<http://www.jmathlib.de/>),

Euler (<http://eumat.sourceforge.net/>)

FreeMat (<http://freemat.sourceforge.net/>) – бесплатные ПО с языком «максимально совместимым» с системой MatLab.

IDL (Interactive Data Language, <http://www.ittvis.com/>) – коммерческое ПО (область применения аналогична системе MatLab, больше подходит для обработки изображений и визуализации).

O-Matrix (<http://www.omatrix.com>) – коммерческое ПО, в котором есть режим совместимости с системой MatLab.

LyME (<http://www.calerga.com>) – ПО, которое позволяет использовать некоторые MatLab-команды на Palm-устройствах.

При составлении заданий и примеров автору очень помогло учебное пособие Питера Аклама [Acklam], а также обсуждения на блоге Лорен Шью [Loren]. Автор позволил себе не указывать у каждого примера источник, поскольку некоторые примеры повторяются на многих сайтах Интернета (и установить их авторство достаточно проблематично). Из русскоязычных источников автору больше всего нравится книга [Потемкин, 1999], которая, возможно, уже немного устарела. В последнее время появляются учебные ресурсы, посвящённые использованию системы MatLab в анализе данных, например [Золотых, MatLab]. См. также информацию сайта [Exponenta.ru].

ЛИТЕРАТУРА, ССЫЛКИ

[[Exponenta.ru](http://www.exponenta.ru)] <http://www.exponenta.ru>

[[Acklam](http://home.online.no/~pjacklam)] *Peter J. Acklam* MATLAB array manipulation tips and tricks // <http://home.online.no/~pjacklam>

[[Loren](http://blogs.mathworks.com/loren/)] *Loren Shure* (блог) Loren on the Art of MATLAB // <http://blogs.mathworks.com/loren/>

[[MathWorks](http://www.mathworks.com)] <http://www.mathworks.com>

[[Griffiths](http://maxwell.me.gu.edu.au/spl/matlab-page/matlab_griffiths.pdf)] *David F. Griffiths* An Introduction to Matlab Version 2.3 // The University Dundee DD1 4HN, 2005. – http://maxwell.me.gu.edu.au/spl/matlab-page/matlab_griffiths.pdf

[[Sigmon](http://terpconnect.umd.edu/~nsw/ench250/primer.htm)] *Kermit Sigmon* MATLAB Primer // <http://terpconnect.umd.edu/~nsw/ench250/primer.htm>

[[Cambridge](http://www-h.eng.cam.ac.uk/help/tpl/programs/matlab.html)] Cambridge University Engineering Department – Matlab // <http://www-h.eng.cam.ac.uk/help/tpl/programs/matlab.html>

[[CVD](http://www.mathworks.com/support/tech-notes/1100/1109.html)] The technical note «How Do I Vectorize My Code?» // <http://www.mathworks.com/support/tech-notes/1100/1109.html>

[[Robbins](http://www.mathworks.com/matlabcentral/fileexchange/2371-good-matlab-programming-practices)] *Michael Robbins* Good Matlab Programming Practices for the Non-Programmer // <http://www.mathworks.com/matlabcentral/fileexchange/2371-good-matlab-programming-practices>

[[Johnson](http://www.datatool.com/prod02.htm)] *Richard Johnson* MATLAB Programming Style Guidelines // <http://www.datatool.com/prod02.htm>

[[ML](http://www.machinelearning.ru)] <http://www.machinelearning.ru>

[[Guide](http://www.mathworks.com/access/helpdesk/help/pdf_doc/matlab/getstart.pdf)] MATLAB® 7 Getting Started Guide http://www.mathworks.com/access/helpdesk/help/pdf_doc/matlab/getstart.pdf

[[Потемкин, 1999](#)] *Потемкин В.* Система инженерных и научных расчетов MATLAB 5.x (в 2-х томах). Диалог-МИФИ. 1999.

[[Золотых, MatLab](#)] *Золотых Н.Ю.* MATLAB в научной и исследовательской работе <http://www.uic.unn.ru/~zny/matlab/>

Mihi ipsi scripsi!