

Занятие 5: Подготовка текстовых отчётов в системе T_EX

Практикум на ЭВМ 2018/2019

Попов Артём Сергеевич

МГУ имени М. В. Ломоносова, факультет ВМК, кафедра ММП

Написание научных текстов

Научные тексты: научные статьи, отчеты, курсовые, выпускные работы и т.д.

Есть правила (стандарты) написания научных текстов.

Зачем нужны правила?

Чтобы разные люди без дополнительных усилий понимали друг друга. Чем жёстче требования к терминологии, языку, форме подаче материала, оформлению, тем быстрее читатель сможет понять основную суть работы.

На этой паре разбираем, как писать отчёты.

У отчёта должен быть заголовок

Из заголовка должно быть понятно:

- ▶ Кем выполнено задание?
- ▶ Какое задание выполнено?
- ▶ К какому курсу относится задание?

Необязательно верстать титульную страницу!

У отчёта должно быть введение

Из введения должно быть понятно:

- В чём заключалось задание?

Введение не повторяет полностью формулировку задания, а лишь кратко описывает её.

Можно ссылаться на формулировку задания, если не получается кратко сформулировать все аспекты.

Введение должно быть конкретным (относится к конкретному заданию, а не к выбранной области анализа данных в целом).

Пример неконкретного введения к заданию

Компьютеры все чаще и чаще выступают помощниками человека. Водитель, потеряв дорогу, скорее воспользуется навигатором, чем спросит путь у прохожего или другого водителя. Захотев связаться с кем-то, мы скорее напишем ему письмо, состоящее из байтов, чем из бумаги и чернил. То же самое можно сказать и о распознавании изображений. Возьмем, как пример, ЕГЭ. Множество учеников со всей России заполняют огромное количество бланков, и все эти бланки необходимо проверить. Что же делать в этой ситуации? Ведь вряд ли кто согласится проверять эти бланки. И здесь на помощь приходят компьютеры. Они распознают ответы учеников и заполняют по ним базу данных, откуда берутся данные для подсчета результата экзамена. Теме анализа изображения, а точнее, анализа рукописного текста, и посвящена данная работа.

Типичная структура отчёта

Структура отчёта примерно соответствует пунктам задания:

1. Введение
2. (опционально) Пояснения к задаче, например выкладки для выведенных в ходе работы формул
3. Список экспериментов
 - 3.1 Дизайн эксперимента
 - 3.2 Результаты эксперимента
 - 3.3 Выводы из эксперимента
4. (опционально) Общие выводы из работы
5. (если есть ссылки) Список использованной литературы

Описание экспериментов: как не надо делать

Задание 2

Векторизованный вариант:

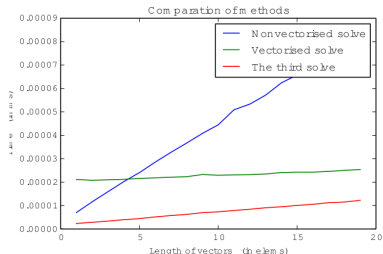
```
def vect_2(x, i, j):
    return np.vectorize(lambda z, y: x[z, y])(i, j)
```

Не векторизованный вариант:

```
def vect_1(x, i, j):
    r = np.array([], x.dtype)
    for k in range(np.size(i)):
        r = np.append(r, x[i[k], j[k]])
    return r
```

Третий вариант:

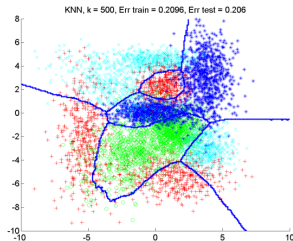
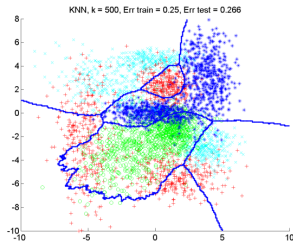
```
def vect_3(x, i, j):
    return np.array([x[t[0]][t[1]] for t in zip(i, j)])
```



Самое оптимальное третье решение.

Описание экспериментов: как надо делать

Метод 500 ближайших соседей сильно недообучен, границы, выдаваемые им, слишком просты. Он обладает высоким *bias* (близкие кривые обучения на высоком уровне ошибки) при низком *variance*. При размере данных меньше 500 (50%) на тестовой выборке наблюдается плато на уровне 75%: классификатор выдаёт в качестве ответа максимальный класс. На тестовой выборке при этом ошибка всё же ниже: сказывается случайный порядок объектов в обучающей выборке, соотношение классов не по 25%, и доминирующий класс составляет чуть более 25%. Однако видна тенденция к падению уровня ошибки. Это происходит потому, что с увеличением объёма данных ослабляется эффект «доминирующего класса», и в этом случае добавление новых данных приведёт к значительному улучшению. Результат для 3000 обучающих объектов будет выглядеть гораздо более приемлемо, а для 5000 он даже приближается к оптимальному:



Выводы проведены.

Анализируйте результаты экспериментов

размер данных	numpy	only python	смешанная
10	1.2	50.21	2.1
100	2.52	40.1	10.2
1000	4.5	45.34	30.95

Что в этих результатах подозрительно?

Анализируйте результаты экспериментов

размер данных	numpy	only python	смешанная
10	1.2	50.21	2.1
100	2.52	40.1	10.2
1000	4.5	45.34	30.95

Что в этих результатах подозрительно?

- Вычисления для размерности 100 и 1000 происходит дольше чем для 10.

Используйте векторные шрифты

Растровые шрифты:

Для каждой задачи написать n вариантов различной эффективности, n — заданный вариант и один вариант работы реализаций на нескольких процессорах. Проанализировать полученные реализации и сделать выводы.

Векторные шрифты:

Для выполнения задания было предложено, для каждой задачи представлены в отдельных python файлах (см. код в папке `tasks`).

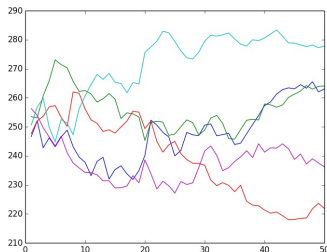
Реализация тестов представлена в модуле `test.py`. Первая часть тестов проверяет корректность решения задачи, вторая часть проверяет скорость решения.

Проведение экспериментов реализовано скриптом `run_tests.py`, направленный на сравнение времени работы реализаций. Эксперимент проводился 100 раз, после чего выбиралось

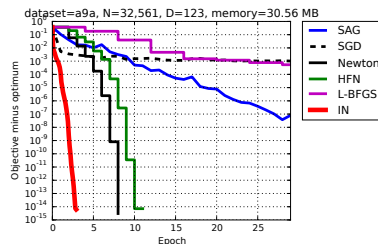
Для генерации текстов с векторными шрифтами достаточно установить пакет T_EX `cm-super`.

Оформление графиков

Плохой график:



Хороший график:



Элементы хорошего графика:

- ▶ Все линии жирные;
- ▶ Есть легенда;
- ▶ По осям указаны значения, сами оси подписаны;
- ▶ По осям выбрана правильная шкала;
- ▶ Сохранён в векторном формате.

Примеры плохо организованных страниц

Таблица 9: Результаты экспериментов задачи N8. Время работы измерено в микро-секундах

	vectorized_8	non_vectorized_8	sy_method_8	multivariate_normal
shape = (50, 100)	940.2	630531.5	42079.1	426.7
shape = (100, 200)	3985.8	4968530.7	386522.0	2636.1
shape = (150, 300)	10492.1	16607857.2	1408267.6	8813.9

Таблица 10: Результаты точности вычисления

	vectorized_8	non_vectorized_8	sy_method_8
shape = (50, 100)	2.29e-13	1.29e-12	3.25e-12
shape = (100, 200)	2.44e-13	1.33e-12	3.29e-12
shape = (150, 300)	2.33e-13	1.29e-12	3.25e-12

Есть большие пустые пространства на странице.

Примеры плохо организованных страниц

Заметим, что стандартная реализация `scipy.stats.multivariate_normal` отстает от `vector_method`:

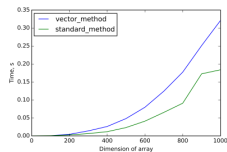


Рис. 6

Погрешность была вычислена как евклидов норм от разности двух функций:

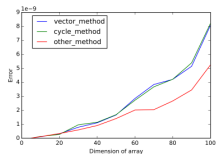


Рис. 7: Погрешность

Графики занимают слишком много места.

Примеры плохо организованных страниц

Измерение времени

Время выполнения функций проверялось на квадратных матрицах X размера $n \times n$ сгенерированных из равномерного распределения и векторах i, j размера n сгенерированных из дискретного равномерного распределения. Результаты (Рис. 2), как и в задаче 1, показывают, что стандартные циклы в Python работают медленнее чем функции и методы, реализованные в библиотеке numru. Индексация в numru массивах работает медленнее чем метод take, но разница небольшая. В отличии от первой задачи, разница во времени выполнения между реализациями с ростом размерности данных не изменяется.

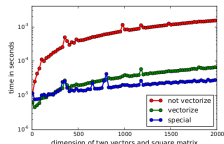


Рис. 2: Зависимость времени выполнения задачи 2 от размерности данных

Есть большие пустые пространства на странице.

Примеры плохо организованных страниц

10 Задача 8

Условие

Реализовать функцию вычисления логарифма плотности многомерного нормального распределения

Входные параметры: точки X , размер (N, D) , мат. ожидание m , вектор длины D , матрица ковариаций C , размер (D, D) . Сравнить с `scipy.stats.multivariate_normal(m, C).logpdf(X)` как по скорости работы, так и по точности вычислений.

Решение 1. Векторизованное

Формула плотности невырожденного нормального распределения выполнена для всей матрицы X .

```
1 def vl_vector(X, m, C):  
2     n = m.shape[0]  
3     ans = -(n/2.0)*np.log(2*np.pi) - 0.5*np.linalg.slogdet(C)[1]  
4     ans -= 0.5*np.dot(np.dot((X-m), np.linalg.inv(C)), (X-m).T)  
5     return np.diag(ans)
```

Текст выходит за поля.

Все числа указываются с необходимым числом знаков

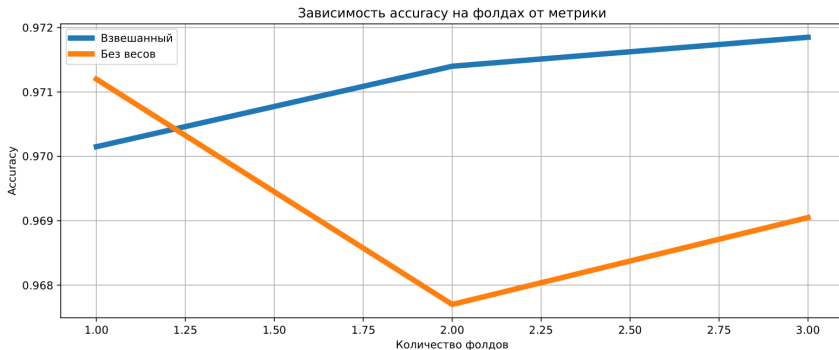
угол (градусы)	fold-1	fold-2	fold-3
0	0.97475	0.9734	0.97375131
5	0.9808	0.9807	0.98160092
10	0.9809	0.98095	0.98170091
15	0.97965	0.979	0.97855107

Таблица 2: ассигасу для разных значений углов поворота

сдвиг (пиксели)	fold-1	fold-2	fold-3
0	0.96175191	0.95740213	0.96340183
1	0.97090145	0.96520174	0.96895155
2	0.96700165	0.96235188	0.96460177
3	0.96565172	0.96295185	0.96355182

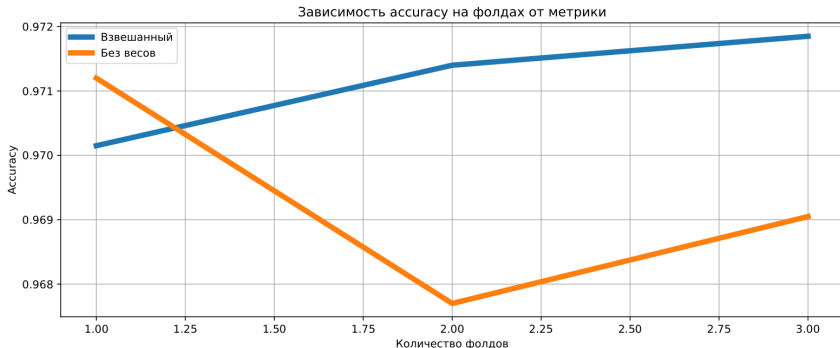
Число указывается с точностью до 2 или 3 знака после запятой.

Думайте об оформлении результатов



Что плохо на этой картинке?

Думайте об оформлении результатов



Что плохо на этой картинке? Почти всё:

- Нет отношения порядка на оси x, опечатки, слишком подробная шкала.

Избегайте длинных предложений

Далее в опытах, для настройки нашего метода, подбора оптимальных параметров, нам придётся использовать кросс-валидацию. Но эта операция занимает очень много времени, поэтому так как все стратегии, перечисленные выше, являются точными, то есть одинаково классифицируют один объект, имея одинаковую обучающую выборку, нам нужно измерить время их работы, и определить лучшую стратегию. Результат опыта на ниже приведенном рисунке.

Чего ещё следует избегать?

- ▶ Ненаучной лексики («результаты модели получились фиговые»)
- ▶ Орфографических ошибок (установите проверку в среде)
- ▶ Грамматических, синтаксических и других ошибок
- ▶ Повествование от первого лица единственного числа
- ▶ Обращений к читателю («вашему вниманию представлены результаты экспериментов»>)
- ▶ Смешения стиля использования буквы «ё» (либо везде используете «ё», либо везде «е»)

Итог: элементы хорошего отчёта по заданию

- ▶ Отчёт подготовлен в системе T_EX;
- ▶ Объём отчёта: 5–20 страниц;
- ▶ Текст отчёта не повторяет полной формулировки задания;
- ▶ Структура отчёта соответствует пунктам задания;
- ▶ Используются векторные шрифты;
- ▶ Графики оформлены надлежащим образом;
- ▶ Шкала для графиков выбрана правильно;
- ▶ На разных графиках результаты для одинаковых методов отображаются одним и тем же цветом;

Итог: элементы хорошего отчёта по заданию

- ▶ Между расположением графиков и местами их упоминания в тексте относительно небольшое расстояние (на той же или на соседней странице);
- ▶ На страницах не должно быть много пустого места;
- ▶ В большинстве случаев графики/таблицы/псевдокоды алгоритмов не должны занимать большей части одной страницы отчёта;
- ▶ Все числа в тексте/таблицах указаны с необходимым числом значащих цифр;
- ▶ В большинстве случаев в отчёте не должно быть никакого кода;
- ▶ Для всех экспериментов описан выбранный дизайн экспериментов, а также сделаны выводы из полученных результатов;

Система T_EX

T_EX — система компьютерной вёрстки, построенная по принципу компиляции документа, записанного с помощью специального языка разметки.

Изобретена Д. Кнудом в конце 70х годов.

Является де-факто стандартом для написания научных статей.

Язык разметки T_EX используется для набора формул во многих системах: в вики-разметке, в matplotlib, Microsoft Office и др.

L^AT_EX — надстройка поверх T_EX.

Дистрибутивы и редакторы T_EX

Дистрибутивы:

- ▶ Windows: MiKTeX, TeX Live
- ▶ Linux: TeX Live
- ▶ Mac OS: MacTeX, TeX Live

Редакторы: WinEdt, TeXnicCenter, Kile, TeXmaker, TeXstudio ...

Режимы компиляции

- ▶ `tex -> dvi -> ps -> pdf`
- ▶ `tex -> dvi -> pdf`
- ▶ `tex -> pdf`

Структура файла .tex

```
\documentclass{article} % класс документа
%Преамбула документа

\usepackage[utf8]{inputenc} % задаём кодировку файла
% задаём правила переносов для русского языка
\usepackage[russian]{babel}

%Текст документа
\begin{document}
Некоторый текст в первом абзаце.
Несколько      пробелов подряд считаются как один.

Конец абзаца задаётся задаётся пустой строкой.
\end{document}
```

Разделы и подразделы

```
\section{Раздел 1}  
\subsection{Подраздел}  
\subsubsection{Подподраздел}  
  
\section{Раздел 2}
```

Формулы

Формула в тексте: $\sin^2 x + \cos^2 x = 1$.

Формула в тексте: $\sin^2 x + \cos^2 x = 1$.

Выносная формула:

\$\$

$A_{ij} = b_i^2 + c_j^3 \quad \forall i, j = 1, \dots, n.$

\$\$

Выносная формула:

$$A_{ij} = b_i^2 + c_j^3 \quad \forall i, j = 1, \dots, n.$$

Формулы

Выравнивание формул:

```
\begin{align}
\tag E &= mc^2 \\
\label{eq:1}&E = mc^2
\end{align}
```

Выравнивание формул:

$$E = mc^2$$

$$E = mc^2 \tag{1}$$

Ссылки

```
\begin{equation}  
\label{eq::1}  
E = mc^2.  
\end{equation}
```

Ссылка на формулу~\eqref{eq::1}

$$E = mc^2. \tag{2}$$

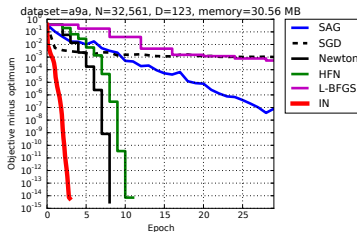
Ссылка на формулу (2)

```
\section{Ссылки}\label{sec:1}  
Ссылка на раздел~\ref{sec:1} в документе.
```

Картинки

Картинка в тексте:

```
\includegraphics[width=5cm]{a9a_epoch.pdf}
```



Картинка в тексте:

Таблица

Таблица:

```
\begin{tabular}{|cc|c}  
  a & b & c \\  
  d & e & f \\  
\end{tabular}
```

Таблица:

a	b	c
d	e	f

Расположение картинок на странице

```
\begin{figure}[h]    %Разместить таблицу здесь
\begin{center}
\includegraphics[width=5cm]{a9a_epoch}
\end{center}
\caption{Картинка}\label{fig::1}
\end{figure}
```

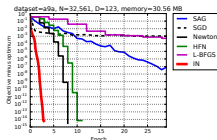
Ссылка на картинку: рис.~\ref{fig::1}

Несколько картинок на странице

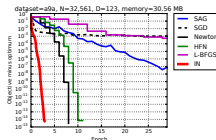
```

\tabcolsep = 20pt %длина разделителя между колонками
\begin{tabular}{cc}
\includegraphics[width=5cm]{a9a_epoch}
&
\includegraphics[width=5cm]{a9a_epoch}\\
(a) & (b)
\end{tabular}

```



(a)



(b)

Особенности типографии: тире и дефис

- ▶ дефисы в словах: из-за δ -функции
дефисы в~словах: из-за `$\delta` δ -функции
- ▶ диапазоны чисел: страницы 3–7
диапазоны чисел: страницы~3--7
- ▶ тире в предложениях: Это — тире.
тире в~предложениях: Это~--- тире.
- ▶ минусы в формулах: $-f(-x) = f(x)$
минусы в~формулах: `$-f(-x)=f(x)$`

Особенности типографии: кавычки

- ▶ Французские «ёлочки»
Французские <<ёлочки>>
- ▶ Немецкие „лапки или 99–66“
Немецкие , ,лапки или 99--66“
- ▶ Английские “лапки или 66–99”
Английские “лапки или 66--99”
- ▶ Неверно: „нигде так не принято”
Неверно: , ,нигде так не принято”
- ▶ Неверно: ”и так тоже никто не делает”
Неверно: ”и так тоже никто не делает“
- ▶ Неверно: "а это вообще не кавычки"
Неверно: "а это вообще не кавычки"

Список литературы

Ссылка в тексте на публикацию~\cite{vorontsovLX}.

% В конце документа

```
\section{Список литературы}
```

```
\begin{thebibliography}{99}
```

```
\bibitem{vorontsovUrl}
```

Воронцов К. В., Полезная информация для
пользователей \LaTeX,

```
\url{www.ccas.ru/voron/latex.html}
```

```
\bibitem{vorontsovLX}
```

Воронцов К. В., \LaTeX в примерах, 2005,

```
\url{www.ccas.ru/voron/download/voron05latex.pdf}
```

```
\end{thebibliography}
```

Список литературы

Ссылка в тексте на публикацию~\cite{blei06}.

% В конце документа

```
\section{Список литературы}
\bibliographystyle{gost780s}
\bibliography{references}
```

В файле references.bib:

```
@ARTICLE{blei06,
author =      {D. Blei and M. Jordan},
title =       {Variational inference for ...},
journal =     {Journal of Bayesian Analysis},
year =        {2006},
volume =      {1},
}
```

BibTeX и русский язык

BibTeX не дружит с кириллицей и utf-8 одновременно!

Способ 1. Сохранить файл с библиографией в cp1251, при запуске предупредить о кодировке (либо вы счастливый обладатель Windows)

```
\inputencoding{cp1251}
\bibliographystyle{abbrv}
\bibliography{liter_cp}
```


BibTeX и русский язык

BibTeX не дружит с кириллицей и utf-8 одновременно!

Способ 2. Использовать при компиляции `bibtexu` вместо `bibtex`
(если вы несчастный обладатель Linux)

Статья[1]

```
\bibliography{liter_utf}
```



П. Артём. [Заголовок статьи.](#)

Журнал, 8(3):305–316, Nov. 2014.

Русский язык в списках

Переопределить счётчики списков второго уровня на русские буквы:

```
\renewcommand{\theenumii}{\asbuk{enumii}}
```

1. внешний элемент списка;
2. другой внешний элемент;
 - 2.а внутренний элемент 1
 - 2.б внутренний элемент 2
 - 2.в внутренний элемент 3

verbatim

Окружение `verbatim` — запрещает LaTeX обрабатывать вставленный текст, отображает код как есть

```
def sum(list_of_numbers):  
    my_sum = 0  
    for elem in list_of_numbers:  
        my_sum += elem  
    return my_sum
```

```
\begin{verbatim}  
def sum(list_of_numbers):  
    my_sum = 0  
    for elem in list_of_numbers:  
        my_sum += elem  
    return my_sum  
\end{verbatim}
```

Пакет listings

Пакет `listings` — мощный пакет LaTeX, позволяющий настраивать специфическое оформление для кода

```
def sum(list_of_numbers):  
    my_sum = 0  
    for elem in list_of_numbers:  
        my_sum += elem  
    return my_sum
```

```
\begin{lstlisting}  
def sum(list_of_numbers):  
    my_sum = 0  
    ...  
\end{lstlisting}
```

Правда, по умолчанию он будет работать почти как `verbatim`...

Пакет listings

```
\usepackage{listings}
\usepackage{color}

\lstdefinestyle{myLatexStyle}{
  basicstyle=\small\ttfamily,
  language={python},
  numbersep=5mm, numbers=left, numberstyle=\tiny,
  breaklines=true, frame=single, framexleftmargin=8mm,
  xleftmargin=8mm, backgroundcolor=\color{green!5},
  frameround=fttt, escapeinside=??, rulecolor=\color{red},
  morekeywords={reduce},
  keywordstyle=\color[rgb]{0,0,1},
  commentstyle=\color[rgb]{0.133,0.545,0.133},
  stringstyle=\color[rgb]{0.627,0.126,0.941}
}
\lstset{style=myLatexStyle}
```

Пакет minted

Пакет `minted` — пакет LaTeX, позволяющий настраивать оформление кода

Плюс `minted` — большое число предустановленных тем

```
def sum(list_of_numbers):  
    my_sum = 0  
    for elem in list_of_numbers:  
        my_sum += elem  
    return my_sum
```

```
\begin{minted}[fontsize=\small]{python}  
def sum(list_of_numbers):  
    my_sum = 0  
    ...  
\end{minted}
```

Пакет minted

Команда `\mintinline{language}{}{}` для оформления кода
внутри связного текста

Команда `\mintinline{latex}{\mintinline{language}{}{}}`
для оформления кода внутри связного текста

Плюсы и минусы различных способов

- ▶ `verbatim`
 - + Быстро, не требует настройки
 - Отсутствие возможностей настройки
- ▶ `lstlisting`
 - + Огромное количество возможностей
 - Красивый результат требует тщательной настройки
 - Сложно задавать свои окружения для разных языков
- ▶ `minted`
 - + Огромное количество возможностей
 - + Больше число предустановленных тем
 - + Легко задавать окружения для разных языков
 - Есть проблемы при установке

При использовании всех этих пакетов объявление слайда приходится записывать так:

```
\begin{frame}[fragile]\frametitle{Плюсы и минусы  
различных способов}
```


TikZ и PFG

PFG — низкоуровневый пакет для векторной графики в T_EX

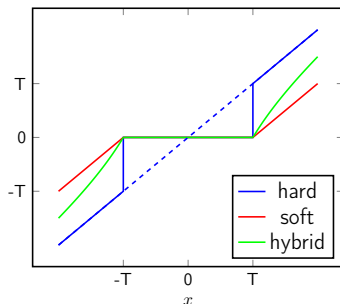
TikZ — высокоуровневое расширение этого пакета

<http://www.texample.net/tikz/> — сайт с примерами работы

<http://www.texample.net/tikz/> — сайт с примерами работы

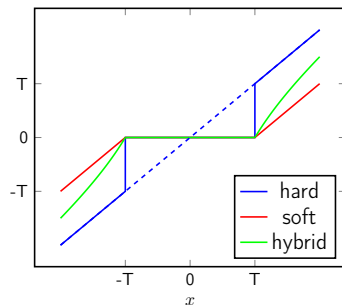
короткая ссылка — самое подробное руководство

Изображение графиков



```
\begin{tikzpicture }[scale=0.7]
\begin{axis}[
  line width = 1pt,
  xlabel =  $x$ ,
  xtick={-1, 0, 1},
  xticklabels={-T, 0, T},
  ytick={-1, 0, 1},
  yticklabels={-T, 0, T},
  mark=none,
  legend entries={hard, soft,
    hybrid},
  legend style={font=\Large,
    legend pos=south east}
]
...
```

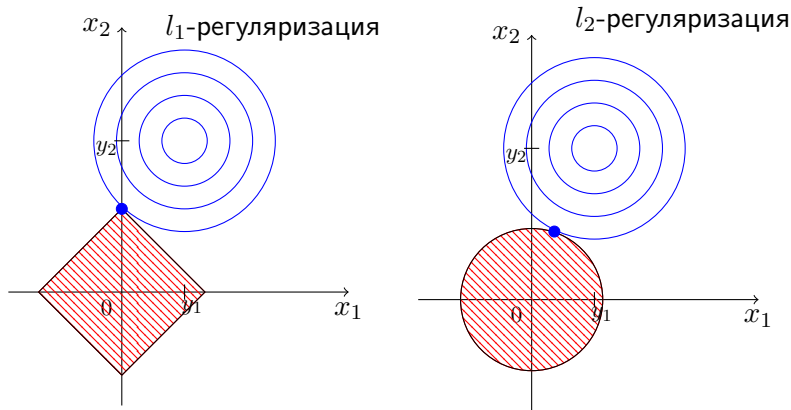
Изображение графиков



```
...
\addplot[blue] coordinates
{(-2, -2) (-1, -1) (-1, 0)
(1, 0) (1, 1) (2, 2)};
\addplot[blue, dashed] coordinates
{(-2, -2) (2, 2)};
\addplot[red] coordinates ...
\addplot[green, domain=-2:-1]
{x - 1/x};
\addplot[green, domain=1:2]
{x - 1/x};
\addplot[green, domain=-1:1]
{0};
\end{axis}
\end{tikzpicture}
```

Изображение линий уровня

Линии уровня для $\|X\vec{w} - \vec{y}\|_2$ и области $\|w\|_1 \leq \kappa$, $\|w\|_2 \leq \kappa$:



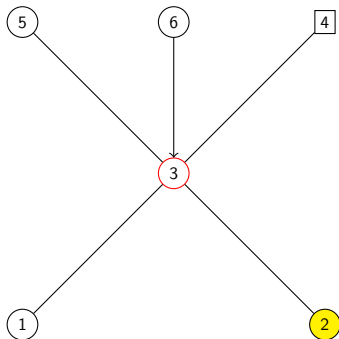
Изображение линий уровня

```
\begin{tikzpicture }
\draw[>-] (-1.5, 0) -- (3, 0) node[anchor=north] {$x_1$};
\draw[>-] (0, -1.5) -- (0, 3.5) node[anchor=east] {$x_2$};
\draw (-0.2, -0.2) node[scale=0.8] {$0$};
\draw (2, 3.5 ) node {$1_1$-регуляризация};

\draw[blue] (0.83, 2) circle (1.2cm);
\draw[blue] (0.83, 2) circle (0.9cm);
\draw[blue] (0.83, 2) circle (0.6cm);
\draw[blue] (0.83, 2) circle (0.3cm);

\draw[red] (0, -1.1) -- (1.1, 0) -- (0, 1.1) --
(-1.1, 0) -- (0, -1.1);
\filldraw[pattern color=red, pattern=north west lines]
(0, -1.1) -- (1.1, 0) -- (0, 1.1) --
(-1.1, 0) -- (0, -1.1);
...
```

Изображение графа

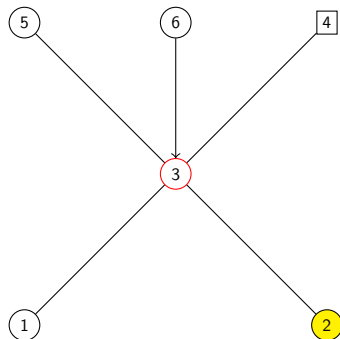


```

\begin{tikzpicture }[scale=2]
\node (p1) at (0,0)
[scale=0.6,shape=circle,
draw=black,fill=white] {1};
\node (p2) at (2,0) [scale=0.6,
shape=circle,draw=black,
fill=yellow] {2};
\node (p3) at (1,1) [scale=0.6,
shape=circle,draw=red,
fill=white] {3};
\node (p4) at (2,2) [scale=0.6,
shape=rectangle,draw=black,
fill=white] {4};
\node (p5) at (0,2) [scale=0.6,
shape=circle,draw=black,
fill=white] {5};
...

```

Изображение графа



...

```
\node (p6) at (1,2) [scale=0.6,
shape=circle,draw=black,
fill=white] {6};
```

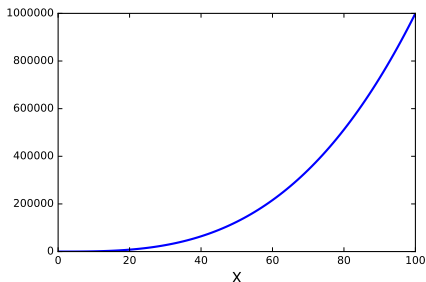
```
\draw (p1) -- (p3) -- (p5);
```

```
\draw (p2) -- (p3) -- (p4);
```

```
\draw[->] (p6) -- (p3);
```

Сохранение графиков экспериментов

Проблема: провели эксперимент, сохранили график, но...



забыли подписать ось y .

Интеграция Python и TikZ

Пакет `matplotlib2tikz` позволяет сохранять графики в формате TikZ.

```
import matplotlib.pyplot as plt
from matplotlib2tikz import save as tikz_save

x = [i for i in range(0, 101, 1)]
y = [x_el ** 3 for x_el in x]

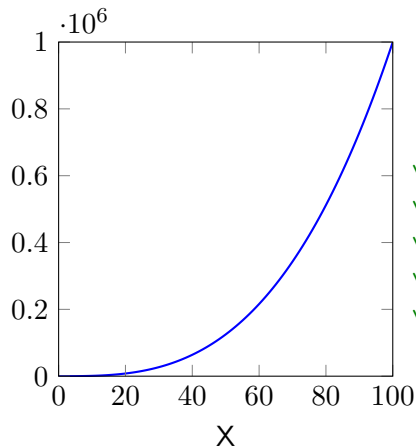
plt.plot(x, y, linewidth=2)
plt.xlabel('X', fontsize=14)

tikz_save('tmp_tikz.txt')
```

Результат сохранения картинки

```
% This file was created by matplotlib2tikz v0.6.13.
\begin{tikzpicture }
\begin{axis}[
xlabel={X},
xmin=0, xmax=100,
ymin=0, ymax=1000000,
axis on top,
tick pos=both
]
\addplot [thick, blue, forget plot]
table {%
0 0
1 1
2 8
...
```

Вставка картинки в формате .tikz



```

\newlength\figureheight
\newlength\figurewidth
\setlength\figureheight{6cm}
\setlength\figurewidth{6cm}
\input{tmp_tikz.txt}

```

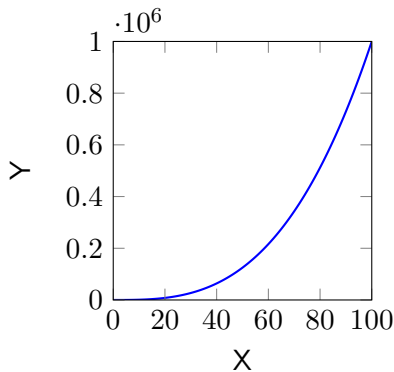
Что это нам даёт?

Вставка картинки в формате .tikz





Картинка сохранена в удобном текстовом представлении

Добавим название оси y отредактировав текстовый файл:

```
...  
\begin{axis}[  
  xlabel={X},  
  ylabel={Y},  
  ...
```



Полезные ссылки

-  Львовский С. М. Набор и вёрстка в системе \LaTeX . 2003.
<http://www.ptep-online.com/ctan/llang2003.pdf>
-  Воронцов К. В. \LaTeX в примерах, 2005,
<http://www.ccas.ru/voron/download/voron05latex.pdf>
-  Написание отчётов и статей (рекомендации), ссылка.
-  Балдин Е.М. LATEX, GNU/Linux и русский стиль, ссылка.