Dr. EW
Johnny Heikell

Scilab для чайников, альтернатива Matlab

Версия 5.3.2 (5.3.3)

Джонни Хейкел

"Было бы частой ошибкой этой стране, чтобы измерить вещи на сумму денег, которую они стоят." Альберт Эйнштейн

Об этой презентации

Я собирал эту презентацию в то время, как познакомился с Scilab для основных инженерных приложений. Узнал, что причина ограниченного распространения SciIab является отсутствие хороших уроков, которые делают обучение SciIab более доступным для программирования новичков. Жаль, потому что Scilab заслуживает более широкого признания. Надеемся, что эта презентация может быть полезной, по крайней мере, в некоторых вопросах Scilab.

Текст, без сомнения, имеет недостатки и ошибки. Я надеюсь придумать улучшенной версии в не слишком отдаленном будущем (с Scilab 6). Пожалуйста комментировать & предложения по адресу:

http://scilabdummies/wordpress.com/

Дань старых богов

Лучшие учебники Scilab не на английском. Ниже приведены те, которые я консультировался наиболее для этой работы:?
Scilab / Xcos учебники Тимо Mäkelä в (3 части) на финском языке <http://sites.? google.com / сайта / tjmakela / главная>. Тяжелый на математического формализма,? стандартный унылый латекс верстки, но лучший, который я знаю?
Arbeiten Жан-Мари Зогг в мит Scilab унд Scicos по-немецки? <http://www.fh-htwchur.ch/uploads/media/Arbeiten_mit_Scilab_und_? Scicos_v1_01.pdf>. Это хорошо, и неофициальный, и содержит детали, которые? Mäkelä опустил. Нуждается в обновлении?
Смешанные учебники Вольфганга Kubitzki на немецком, которые можно найти на? <http://www.mst.fh-kl.de/~kubitzki/>. Довольно хорошо, много деталей, мало? практические примеры (скрипты в отдельные файлы. ZIP)

Я использовал их в работе.

«Чтобы скопировать из одного -плагиат, копировать из многих является исследование». Неизвестный

Почему я сделал так, как я это сделал

Как аспирант на КУ в 1990-91 годах, мне нужно было быстро узнать MathCAD или Matlab.Однокурсник показал мне MathCAD основы в течение 15 минут с использованием синусоидальной функции. Лекция-то вроде этого:

"Прежде всего, объявите переменные, которые вам нужно"
"Тогда вы определите функцию, которую вы хотите построить"
"После этого вы пишете команды сюжета"

С этого учения я начал и был в состоянии использовать MathCAD для моего MS диссертации.

Извлечение из урока: Показать примеры и пропустить академической мелочи.

Я глубоко благодарен Джиму за его уроки. Мы будем повторять его, как только установим и откроем Scilab.

Почему PowerPoint?

Почему я выпускаю этот учебник в виде презентации PowerPoint *, когда есть достаточно материала для 400-страничной книги? Есть несколько причин:

- Они изначально были личными заметками, я узнал только потом, что они могут быть использованы для других целей;
- Легко редактировать PPT;
- Вы получаете быстрый обзор лекций под рукой, выбирая нужный слайд
- PPT имеет преимущество перед PDF в позволяет читателю изменить работу так, как ему(ей) нравится;
- Вы можете копировать-вставить, предусмотренные скрипты в редакторе Scilab без необходимости переписывать их, необходимы лишь незначительные редактирования;

***). Документы п.п. не требуют программного обеспечения MS. LibreOffice работает, а также (по крайней мере, до PPT 2003), но некоторые редактирования могут быть необходимы. Oracle выбросил полотенце на OpenOffice в апреле 2011 года, но она живет в Apache инкубатор.**

Почему имитации?

Британский физик и инженер лорд Кельвин (Уильям Томсон), как известно, сказал: "Когда вы можете измерить то, что вы говорим о и выразить его в цифрах, вы знаете что-то о этом ".
Его слова можно перефразировать в терминах компьютер возраста: «Когда вы можете имитировать то, о чем вы говорите о и представляете визуально, Вы что-то знаете об этом»

Лорд Кельвин 1827-1904

Содержание

Dr. EW
Johnny Heikell

1. Введение

Что такое и зачем использовать Scilab?

Что Scilab представляет собой (1/2)

- Пакет программного обеспечения для научных и инженерных вычислений, очень похож на Matlab;
- Scilab является инструментом для числовых вычислений, как и Excel, GNU Octave, Matlab и т.д. В качестве альтернативы можно символьных вычислений, к которым относятся Maple, MathCad, Mathematica, и другие;
- Разработано Консорциума Scilab (DIGITEO), за которой ряд французских институтов и компаний;
- Включенные в пакет Scilab является Xcos, графическое моделирование и моделирование инструмент. Тем не менее, он не совместим с Simulink. Xcos 1.0 пришел с Scilab 5.2, до этого был Scicos. Путаница в комплекте с соперником называемой Scicoslab;
- Scilab является бесплатным и может быть загружен с www.scilab.org

Что представляет собой Scilab is (2/2)

- Scilab является матрицо-ориентированной, так же, как Matlab;
- Это позволяет делать матричные манипуляций, 2D/3D черчение, анимации и т.д;
- Это открытая среда программирования, которая позволяет пользователям создавать свои собственные функции и библиотеки ;
- Ее редактор имеет встроенный, хоть и элементарный, отладчик
  Основными компонентами Scilab являются:
  переводчик
  Библиотеки функций (процедуры, макросы)
  Интерфейсы для Fortran, Tcl / Tk, C, C + +, Java, Modelica и LabVIEW, но не для Python и / или Рубин

Что "лучше", Matlab или Scilab?

Matlab превосходит Scilab во многих отношениях, но Scilab догоняет. Использование Matlab мотивируется только в особых случаях из-за его высокой стоимости

Зачем использовать Scilab? Личные причины

- Matlab 6.5 (R13) не был совместим с моим новым ноутбуком Windows Vista. MatWorks, Inc, рекомендуется покупать новую версию.
  Я отказался заплатить другой лицензионный сбор за Matlab и пошел искать альтернативы с открытым исходным кодом:
- *Мудрец* чувствовал громоздкие, незрелые, и сосредоточился на чистой математике;
- *Python* не оптимизирован для научных и инженерных задач;
  Python (x, y) испортил мой компьютер, когда я установил его. Может быть, я должен я попытался SciPy вместо этого?
- Я устал от GNU Octave, прежде чем я понял, как загрузить и установить его (я хочу инструмент для использования, а не борьбы с ним) .
- Scilab был пятым альтернатива, которую я посмотрел. Это не дало насущные проблемы, так что я застрял в нем. Позже я сталкивался с ошибками и сбоями / зависаниями и разочаровываюсь в своей плохой документации.

Буду ли я все еще выбрать Scilab? Да, я нахожусь под впечатлением Scilab и считают, что другие программы приведут к седым волосам, так или иначе.

Почему люди не используют Scilab

Ниже приведены некоторые комментарии о Scilab и открытого программного обеспечения в целом, с какими я столкнулся:

"Scilab? Никогда не слышал о нем "
"Октава ближе к Matlab"
"Как компания, мы должны использовать программное обеспечение, которое будет поддерживаться лет через десять? "
«Он не имеет наборы инструментов, которые нам нужны"
"Существует расходы, связанные в переходе на новый инструмент программного обеспечения, даже если инструмент бесплатный "
«Обучение и документация-поддержка бедных"
"Там нет интерфейсов для других программных средств, которые мы используем"
"Казалось бы, довольно медленно"

Вывод: Scilab, как и другие программы с открытым исходным кодом, не вызывает доверия в глазах пользователей, особенно профессиональных пользователей. Аналогичная ситуация и с различными превосходными дистрибутивов Linux и офисного пакета LibreOffice. Пользователи доверяют продуктам, которые должны быть оплачены.

Преимущества Scilab

- Цифровая вычислительная лучше подходит для сложных задач, чем символьных вычислений;
- Не все математические задачи закрыли формы решения, цифровые вычисления будут всегда необходимы;
- Scilab похож на Matlab и постоянно развивается еще ближе. Это довольно легко для перехода от одной программе, к другой;
- Scilab требует меньше дискового пространства, чем Matlab и GNU Octave Она включает в себя переводчик Matlab-на-Scilab.
- Построение данных, как говорят, проще, чем с GNU Octave (но наблюдается тенденция к более сложной структуре);
- Xcos инструментов автоматически устанавливает с Scilab, будь то, что Xcos не совместим с Simulink
- Scilab устанавливается без неотложных проблем на компьютерах Windows;
- Scilab является свободной. Борьба за ограниченного числа дорогих лицензий (Matlab, Mathematica и др.) не является проблемой в профессиональной жизни.

Недостатки Scilab

- Цифровая вычислительная вводит ошибки округления, вопреки символьных вычислений;
- Обучение необходимое для числовых вычислений, сложнее, чем для символьных вычислений;
- Для Scilab отсутствует единый учебник /руководство пользователя. Вы "попробовали и плакали" и тратить время на поиск информации о его использовании *
- В некоторых случаях Scilab работает гораздо медленнее, чем Matlab и GNU Octave (улучшения, как говорят, в процессе выполнения);
- Инструменты Scilab для создания графических интерфейсов являются бедными по сравнению с Matlab;
- Помощь Браузер очень формальна и мало полезна для новичков;
- Scilab содержит ошибки и имеет тенденцию к краху;
- Онлайн поддержка со Equalis стоит $ 495 в год и даже больше.

***) Scilab не одинок. Община с открытым исходным кодом имеет плохую репутацию в документации, потому что "документы" не приносит признание.**

Терминология: "функция"

Язык программирования принес путаницу с его неограниченным использованием термина «функция», и это повторяется в Scilab. Этот термин относится к (по крайней мере):

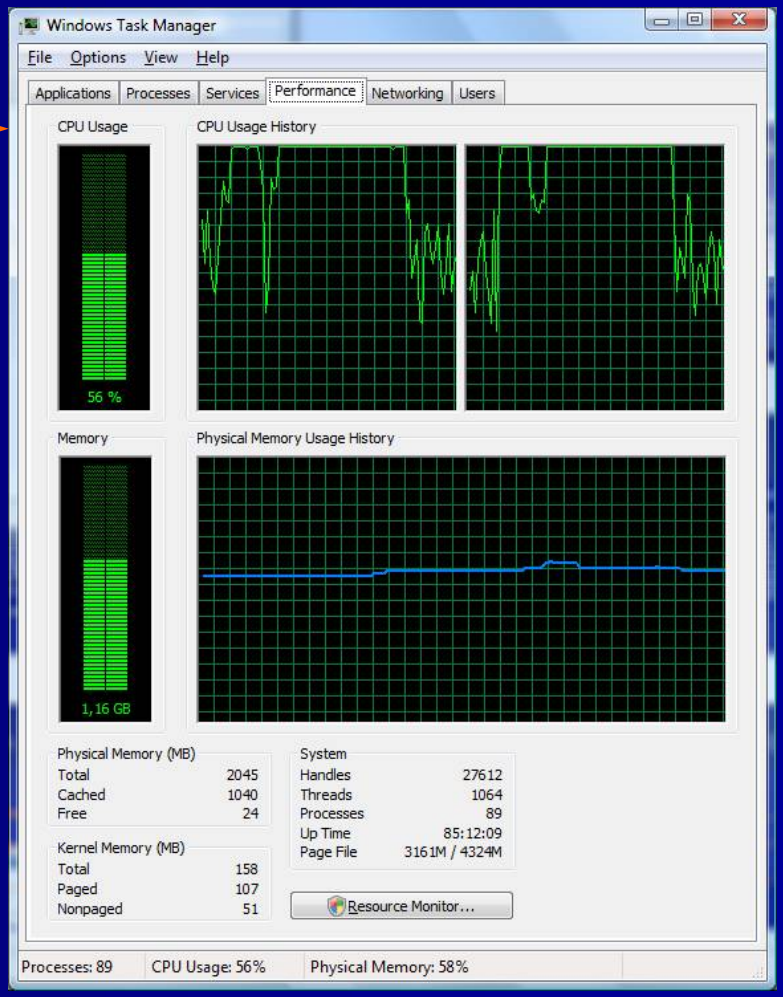
- Математическим функциям в целом
- Встроенным функциям Scilab
- Пользовательским функциям (UDF)

Я бы предпочел, термину функции, макрос (или процедуры), и подпрограмма соответственно . Иногда я говорю о подпрограмме, но это не всегда возможно. Например, функция это термин, который должен быть использован для определения пользовательской функции в Scilab. И есть также риск добавляя к растерянности, применяя собственную терминологию. Путаница остается ...

Введение в проблемы (1/3): Сбой и зависаний

Процессор нагрузки такого масштаба является нормальным во время загрузки компьютера. Однако, это ситуация во время работы Scilab, и я закрыл его. "WScilex.exe" был еще один из его зависания и должны быть закрыты с помощью диспетчера задач (или перезагрузкой компьютера).

Стандартный ответ команды Scilab к проблемам, это введение вкопмьютер последней даты и времени. Scilab не работал в моем ПК Windows Vista.



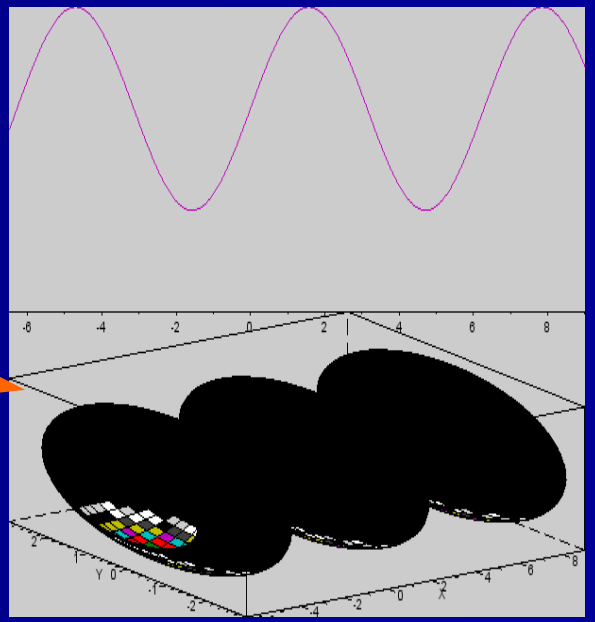
Введение в проблемы (2/3): Новинки *

- В Scilab 5.2 пришел с проблемой, какой я не испытывал с версии 5.1.1: Копировать-вставить из редактора Scilab для PowerPoint часто влечет за собой зависание. Ошибка была исправлена;
- В Scilab 5.3.0 я нашел, что пути Файл / Открыть файл.. и Файл / Сохранить файл .. на редакторе были безразличны;
- Некоторые сценарии, которые я первоначально написал с помощью Scilab 5.1.1 не работал в Scilab 5.3.0, и это является настоящей головной болью;
- Обычно крупные обновления приходят с ошибками;
- Поэтому держите старую версию Scilab пока не доверитесь,, что новой версии можно доверять .

***) Различные варианты Scilab упоминаются. Я работал с Scilab 5.1.1 - 5.3.2. Scilab 5.3.3 пришли слишком поздно, чтобы быть рассмотрены.**

Введение в проблемы (3/3): атомы и умники

- Атомная система Scilab для загрузки и установки пользователя, разработана наборами инструментов. Это дало мне реальное седые волосы?
  Я установил два набора инструментов и Scilab участки стал беспорядок. Здесь вы можете увидеть то,как позже выглядели поверхности вращения с установленным набором инструментов?
  Я обнаружил, его причину после переустановки Windows, и наборов инструментов.
  Идея взносов пользователей в основном звук, но есть риск у ботаников, которые имеют больше рвения, чем способности и упорство, чтобы должным образом проверить свои программы.



Возможность размещения информации.

Scilab поставляется с некоторыми встроенными информационными структурами. Основными из них являются:

- Помощь Браузера, которая может быть доступна из различных окон. Его полезность улучшена в Scilab 5.3.1 когда были включены демонстрации, но Помощь Браузер по-прежнему крепкий орешек для новичков. Это путает, иногда ссылаясь на устаревшие функции;
- Демонстрации, которые могут быть доступны из консоли, не совсем уроки и некоторые из них действуют смешно, некоторые из них могут привести к Scilab зависанию.
- Сообщения об ошибках отображается на консоли. Основные сообщения, иногда запутанны, иногда слишком кратки;
- Что на самом деле хватает, так это встроенного учебника (или даже руководства пользователя в стиле Matlab), которая обновлялась бы с каждым выпуском Scilab.

Информация в Интернете (1/2)

- **Главный портал является Wiki Scilab**, <http://wiki.scilab.org/Tutorials>, были большинство доступных учебников перечислены
  Кузница <http://forge.scilab.org/> Scilab является хранилищем "незавершенного", многие из которых существуют только на бумаге. Его набор проектов документов является ценным;
- **Бесплатные сайты:**
- Сайт Exchange <http://fileexchange.scilab.org/>. Новый дискуссионный форум под управлением команды Scilab, позволяющая легко обмениваться файлами, сценариями, данными, опытом и т.д."
- Дискуссионная группа Google в <http://groups.google.com/group/comp.soft-sys.math.scilab/topics>
- MathKB <http://www.mathkb.com/>. Содержит, среди прочего, форум Scilab.
- В основном продвинутые вопросы произносимого учебник <http://spoken-tutorial.org/Study_Plans_Scilab/>. Видеоуроки строящиеся по IIT Bombay, основы Scilab.

Информация в Интернете(2/2)

- YouTube имеет некоторые видеоклипы о Scilab, но ничего особо ценного;
- Equalis <http://www.equalis.com>. Регистрируясь, Вы получаете бесплатный доступ к дискуссионному форуму;
- <http://usingscilab.blogspot.com/> раньше был очень хороший блог, но теперь неизлечимо болен. Стоит проверить материал, который все еще там.
- Scilab Индия <http://scilab.in/> в основном зеркало Scilab Wiki, с добавлением устаревшего материала и менее активным форумом обсуждения;
  Если вы знаете немецкий:
- Немецкие технические колледжи производят полезные учебники по основам Scilab (лучше, чем у французов). Поиск в Интернете, например, используя термины "Scilab" + "Einführung" и ограничить возможность языка на немецкий.

Вывод: много ресурсов ушли в производство существующей рассеянной документации, но они были несогласованны и внесли усилий. Паршивый менеджмент!

Книги

Существует не один хороший учебник на английском языке на Scilab , как вы найдете в изобилии на Matlab. Это книги , с которыми я знаком:

Бэатер.П. : Технология управления и технологии моделирования с Scilab , 2010. Основные системы управления инженерной-механики. Scilab играет лишь незначительную роль в книге.

Дас, В.В. : Программирование в Scilab 4.1, 2008. Справочное руководство с непривлекательным макетом, устаревших функций, но бесполезное из-за отсутствия практических примеров.

Чансселир Ж.-П. и др. : . Введение в Scilab, издание Deuxieme, 2007 года. Средний учебник с некоторой техникой приложений. Немного устаревший.

Кэмпбелл, С.Л. и др. : Моделирование в Scilab / Scicos , Springer,? 2006 . На основе Scilab 3.1 , более половины книги о Scicos.

Гомес, С. и др. : . Инженерные и научные вычисления в Scilab, Birkhäuser , 1999 . Часто упоминается, но устаревший и не имеет смысла.

Об обновлениях и литературе

Scilab быстро развивается и часто встречаются устаревшие функции . Функции часто объявлены устаревшими, хотя Scilab все еще может поддержать их, и другие функции вообще убрать. Там, очевидно, не коллекция устаревших, а их текущие эквиваленты (если таковые имеются).

Команда Scilab медлит с информацией о глобальных обновления. Например , графический интерфейс , как говорят , был полностью обновлен с версии 5 , но до сих пор я видел описание графического интерфейса только для 4х версий. А прошло уже почти три года …

Бурное развитие является причиной того, почему ограничена литература по Scilab. По большей части устарелое, иногда откровенно вводит в заблуждение . Я получил практический опыт со всеми изменениями , которые должны были быть сделаны , для 5.1.1 , прежде чем они согласились работать на версии 5.3.x .

Препятствия в обучении Scilab

Обучение Scilab может быть неприятно для лица с небольшим опытом программирования. Самыми большими препятствиями являются:

- Отсутствие практических обучающих программ для новичков на английском языке. Ситуация лучше, хотя не намного, с некоторыми другими языками.
- Чрезмерное количество функций Scilab. Их существует около двух тысяч. Часто мы стоим перед выбором, какую функцию использовать, какая работает, а какая - нет.
- Бесполезная Помощь Браузера. Даже тогда, когда у вас есть подозрение, какую из функций использовать, вы не можете получить его прямо из-за загадочного объяснения Помощь Браузера.
- Основные ошибки программирования. Создание бесконечные циклы, деления на ноль, и тп.

На положительной стороне ...

- Scilab работает! Несмотря на мои жалобы она в основном делает прекрасную работу.
  Это великая вещь, что оно дается даром для всех нас, кто не может позволить себе дорогие инструменты для коммерческого моделирования.
  Это великая вещь, бесплатная для всех коммерческих и некоммерческих организаций, которые заботятся о рентабельности.
  Это бесплатный подарок (хотя и с ограничениями *) в области науки и техники и заслуживает нашу поддержку, кто счастливо скачивает все, что приходит бесплатно в Интернете.
  Он заслуживает поддержки, поскольку Scilab, как и другие с открытым исходным кодом ИТ-решений, предстоит тяжелая борьба с обширными коммерческими интересами скептических лиц.
  Да здравствует свободное сообщество!

Dr. EW
Johnny Heikell

2. Первый взгляд на Scilab

С чем вы сталкиваетесь при попытке начать, в том числе "Scilab за 15 минут"

Установка Окна (1/3)

1 . Скачать Scilab от www.scilab.org (Windows на вершине, другие ОС ниже)

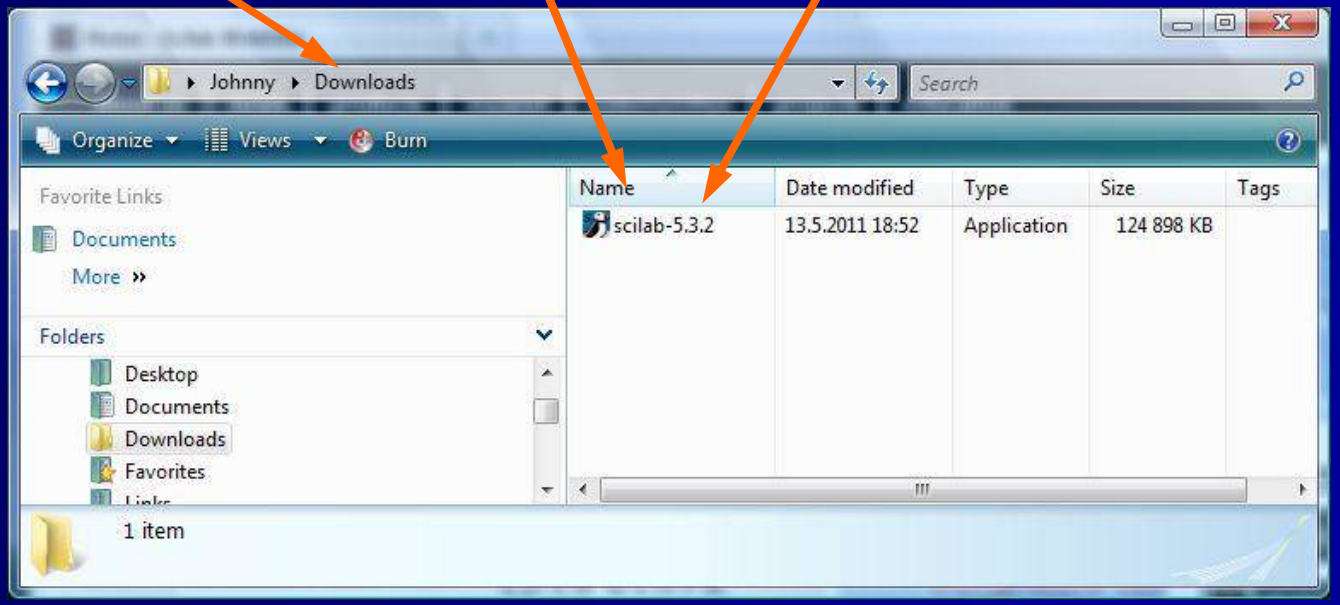
2. Нужная операционная система должна быть сверху. Сохраните файл, как правило, он идет к вашей собственной папке Downloads.

Установка Окна (2/3)

Внутри файла Загрузки

3. Сканирование загруженный файл на наличие вирусов

4. Дважды щелкните на файл, чтобы установить Scilab, следуйте инструкциям на экране

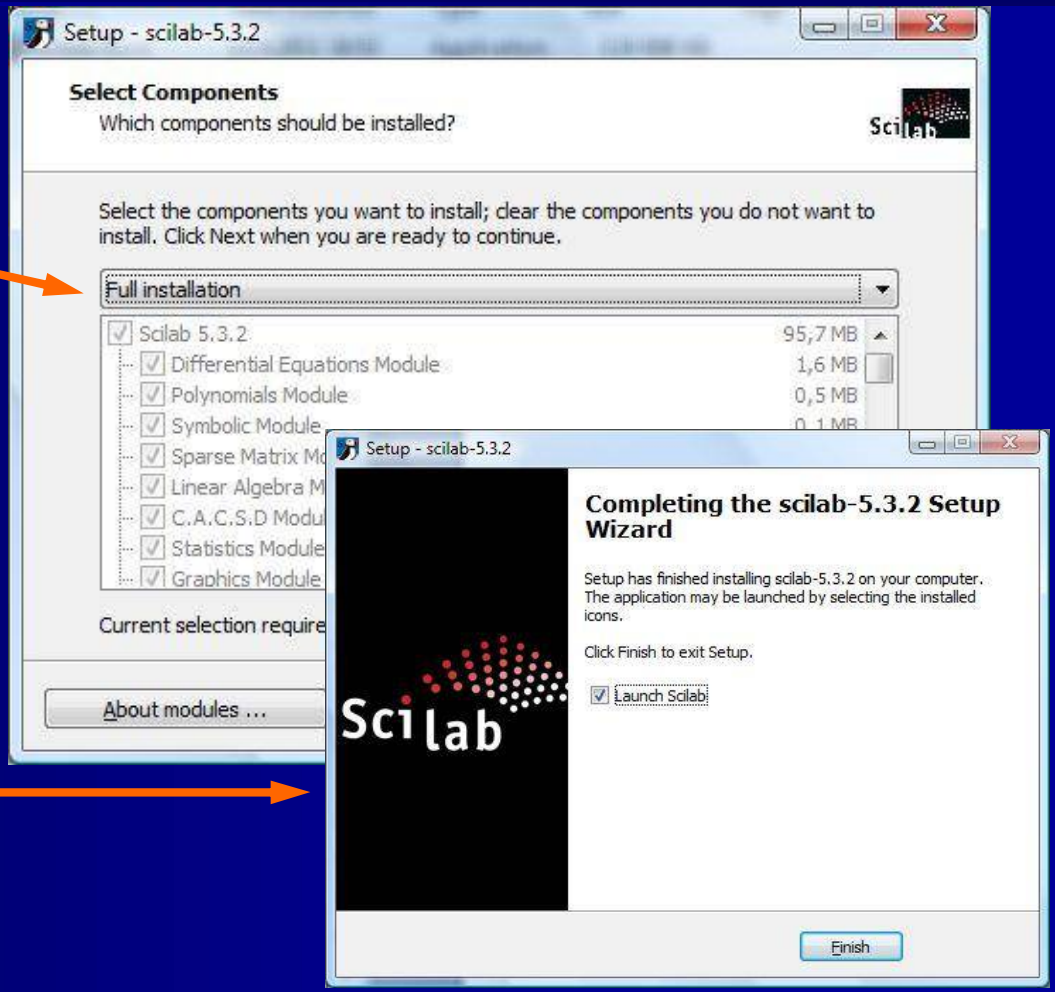


Установка Окна (3/3)

5. Scilab позволяет предположить, что она должна установить все наборы инструментов (модулей). Пойти на это, если вы не действительно хватает памяти

6. Принять условия лицензии (у вас нет другого выбора, так почему бы они просят?). Нажмите кнопку Далее столько раз, сколько необходимо.

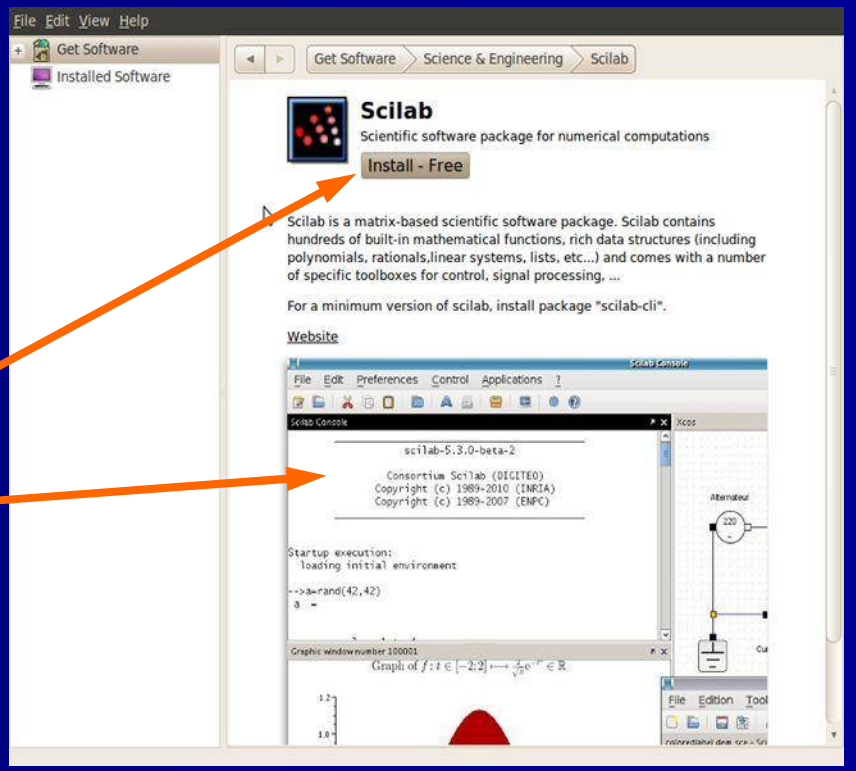
7. Все готово для использования Scilab (нет необходимости перезагрузить компьютер)?
Примечание: Scilab не удалит старую версию



Установка для Linux

Это обсуждение действительно для Ubuntu 10.04 LTS с рабочего стола GNOME *

- Нажмите: Приложения / Ubuntu Software Center / Наука & Engineering и прокрутите вниз до Scilab, а затем просто нажмите кнопку Установить?
  Только Scilab 5.3.0 бета-2 доступен в репозитории?
  Для получения последней версии необходимо перейти на веб-сайт Scilab и скачать бинарные файлы Linux. Установка, однако, является более хитрым вопросом, и я не раскрываю его здесь (не пробовал).



***) Ubuntu 11.04 с Unity была выпущена, но я не пошел на это**

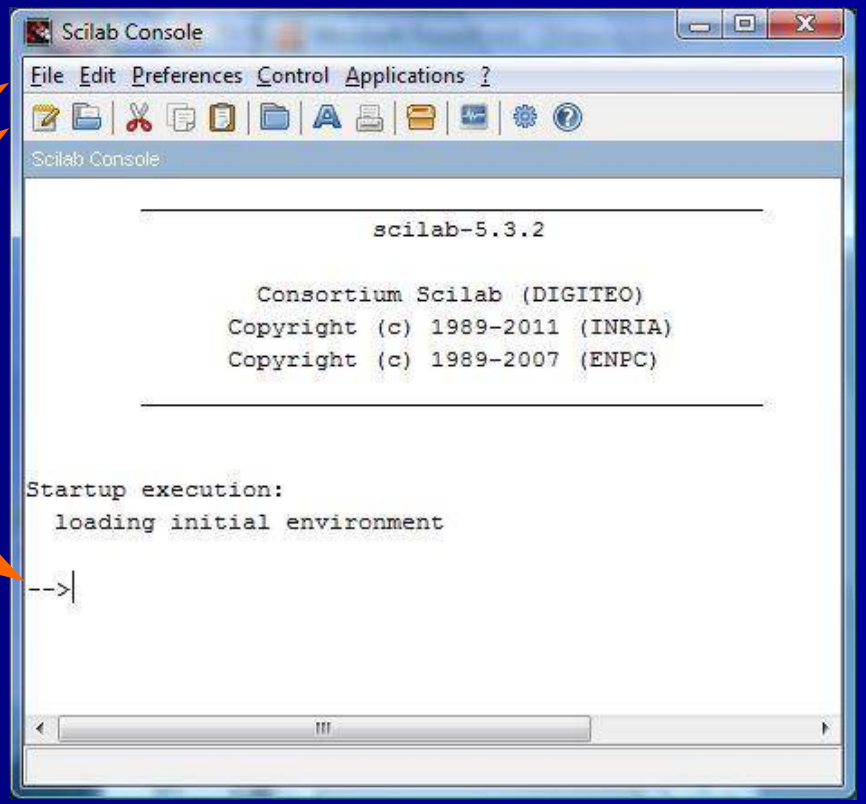
Консоль

Нажмите на значок ярлыка Scilab, чтобы открыть консоль (Command Window в Matlab *):

Строка меню

Панель инструментов

Командная строка



Если ярлык не был создан: Нажмите: Пуск \ Все программы \ Scilab \ Scilab (не выбирайте Scilab консоль)

Люди:

Здесь речь идет, о том, что дал на уроке по MathCad, Джим еще в 1991 году, преобразованная в Scilab. Золото в трех слайдах.

Scilab за 15 минут(1/3): написать сценарий

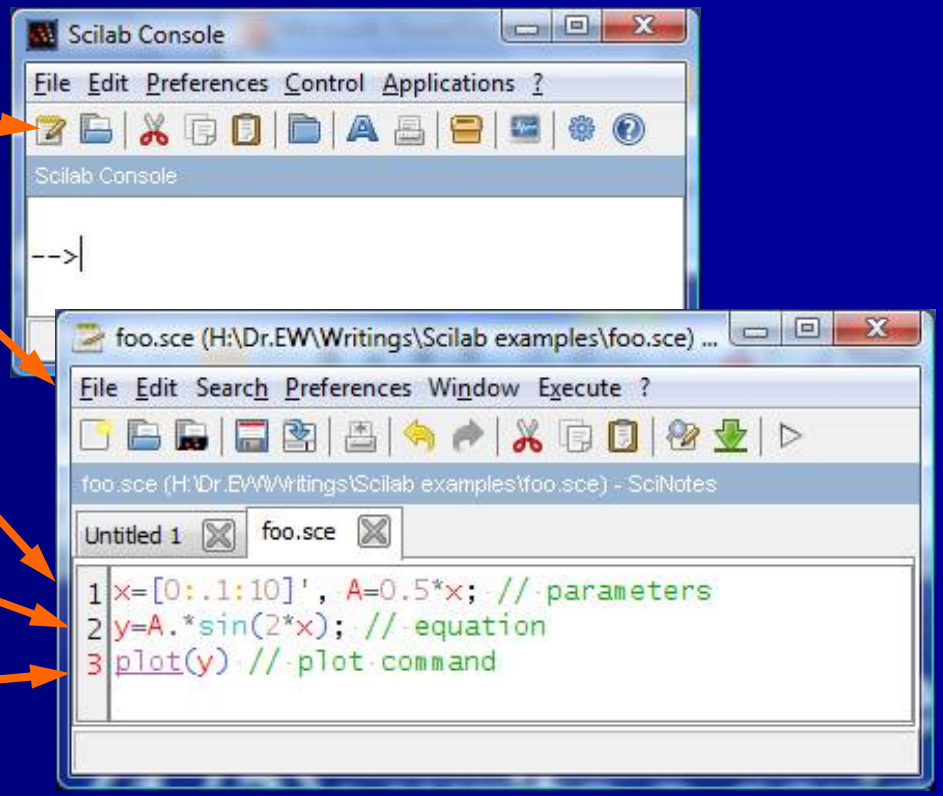
Вспомнив, как Джим научил меня MathCAD за 15 минут? Теперь мы будем повторять этот урок в Scilab. Мы делаем это с помощью редактора (SciNotes):

Шаг 1: На консоли выберите самый левый значок на панели инструментов. Редактор всплывает

Шаг 2: Определить все переменные ваша функция должна (строка 1). Примечание комментарий (/ / ...)

Шаг 3: Далее, определим функцию (синус) в случае (строка 2)

Шаг 4: Наконец, напишите команду (строка 3)

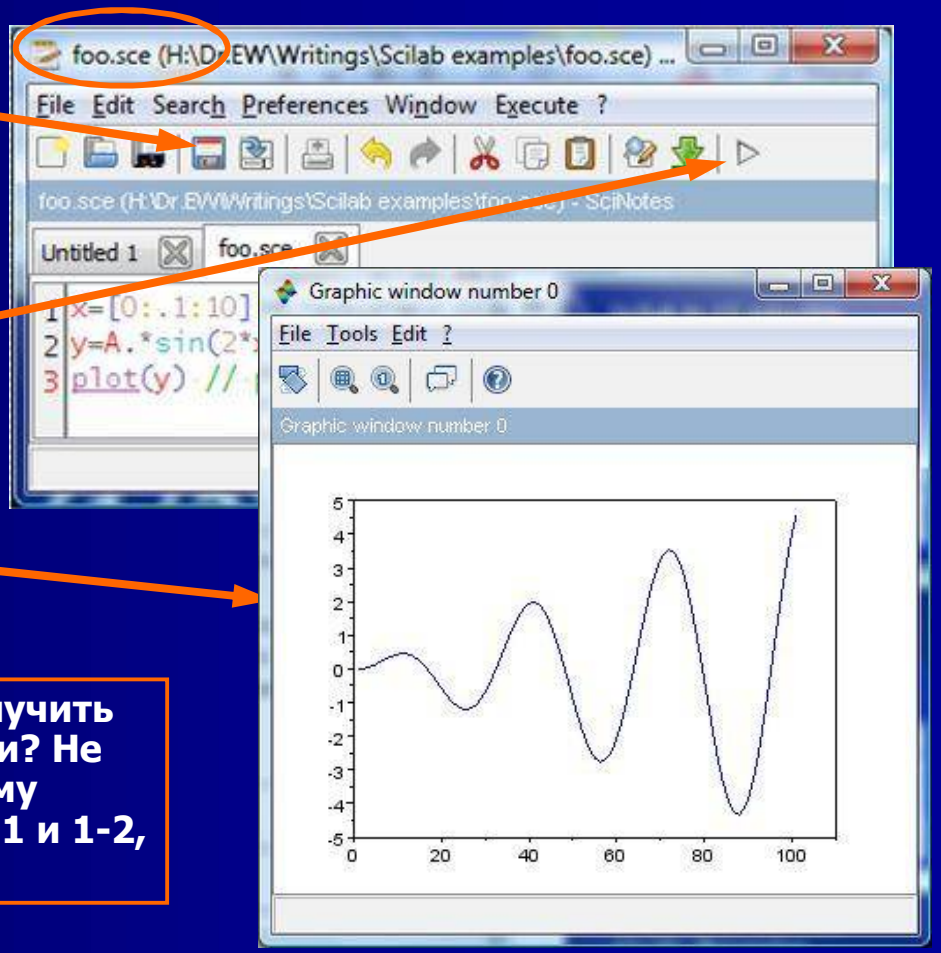


Scilab за 15 минут(2/3): написать сценарий

Шаг 5: Сохраните сценарий, нажав на иконку «Сохранить» и назовите ее, например «foo.sce»

Шаг 6: Взаершение, запустив (выполнения) сценарий, щелкните значок Выполнить (второй шел с 5.3.2)

Шаг 7: Всплывает графическое окно с участком определенного уравнения



Было ли у вас проблемы или получить сообщение об ошибке на консоли? Не волнуйтесь, мы вернемся ко всему позже. Перейдите к примерам 1-1 и 1-2, если вы в спешите.

Scilab за 15 минут(3/3): дискуссии

Этот анализ выявил первой необходимости Scilab в инженерных приложениях:

- Пользовательский интерфейс Scilab состоит из трех основных окон: Консоль, которая появляется, когда Scilab открывается и на котором он выводит текстовые данные (числовые ответы, сообщения об ошибках и т.д.);
- Редактор (SciNotes), которая является основным инструментом для записи, сохранения и выполнения сценариев (программ);
- Графическое окно, в котором Scilab представляет графики;

Рецепт использования Scilab, которому Джим научил меня:
Сначала вы объявляете переменные, которые необходимы.
Тогда вы определить функцию, которую вы хотите построить
И, наконец, подключите в инструкции участок.

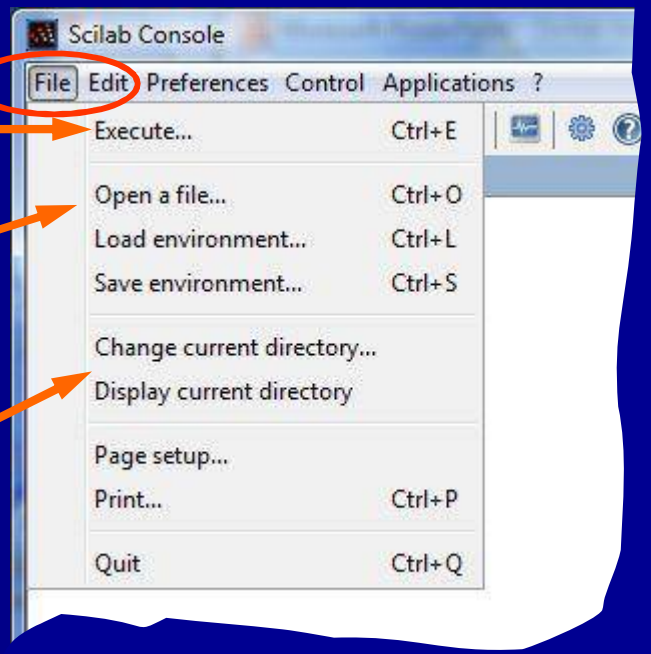
Строка меню консоли(1/6): Файл

Среди функций в соответствии с файлом выпадающего меню, вы столкнетесь:

Выполнить ...: Отсюда вы можете запускать сценарий Scilab (или из редактора, как видно из дальнейшего)

Открыть ...: Как и в Открытом ... Команда в программах MS Office

Изменить текущий каталог ..., отображение текущего каталога: Обратите внимание на предыдущих командах будут необходимы адреса, чтобы сообщить Scilab, где искать сценарий, который вы хотите открыть.



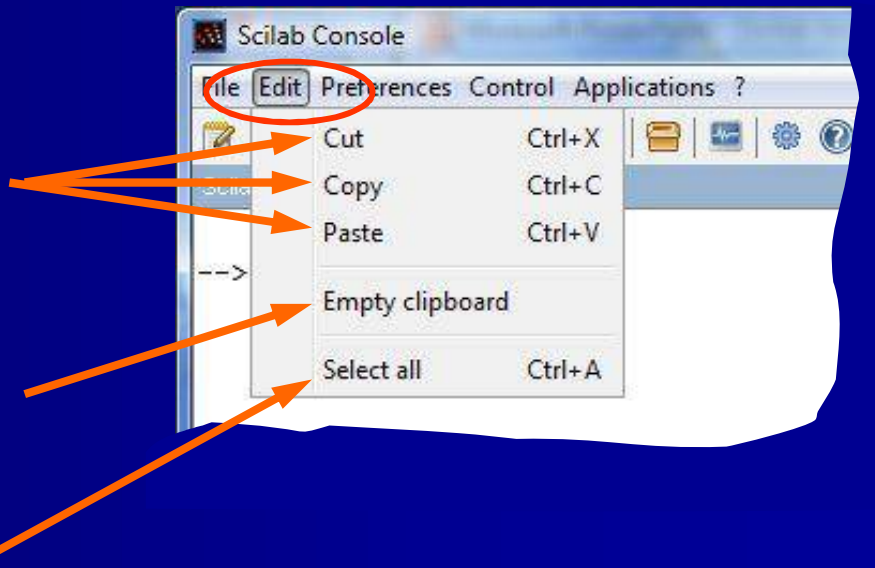
Строка меню консоли(2/6): Редактирование

Функции в раскрывающемся меню Правка говорят сами за себя.

Вырезать, Копировать, и Вставить имеют свои собственные иконки в панели инструментов. Вы также можете найти их, щелкнув правой кнопкой на ПК мыши

Будьте осторожны с пустым буфером обмена. Вы не сможете использовать функцию Копировать после нажатия на него! (Случилось со мной)

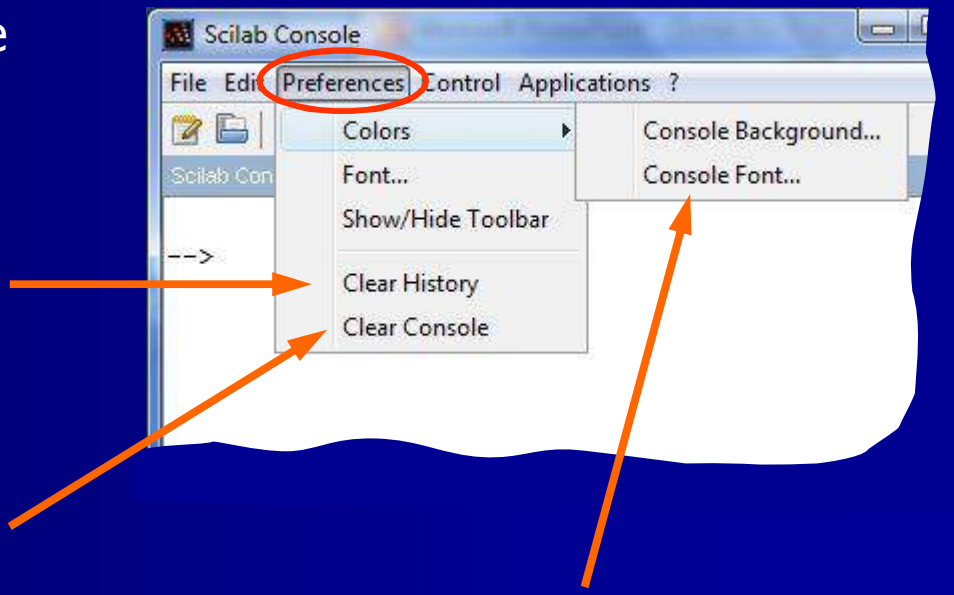
Я использовал Выделить, скопировать-вставить, демонстрировать, в данной презентации



Строка меню консоли(3/6): Предпочтения

Функции под Preferences в раскрывающемся меню очень похожи на то, что вы можете найти на ПК

Я могу только догадываться, что Очистить историю похоже на Удалить личные данные в Firefox, но нет. Показать историю альтернативой Помощь- бесполезно

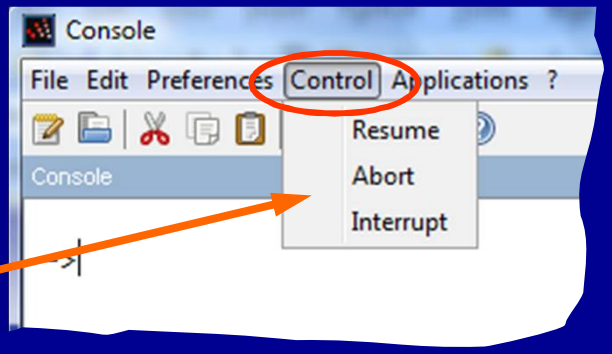


Изменение внешнего вида консоли

Строка меню консоли(4/6): Control

Мне не нужно в раскрывающемся меню управления ни разу при этом эту презентацию, поэтому, очевидно, это не очень полезно

Мое предположение было бы, что альтернативы резюме, прервать, и прерываний дать пользователю возможность вмешиваться в выполнении программы



Помощь Браузер не очень полезно, и это даже не опознает команду прерывания

Строка меню консоли(5/6): Применения

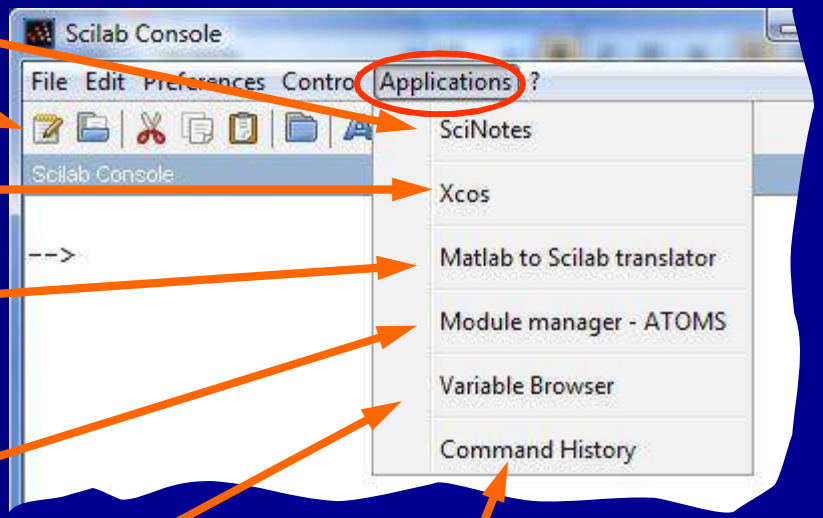
SciNotes: Открывает текстовый редактор Scilab в (так же, как старт SciNotes в панели инструментов)

Xcos: Открывает Xcos

Matlab в Scilab переводчика:.. Использовано перевести Matlab м-файл в Scilab файл.

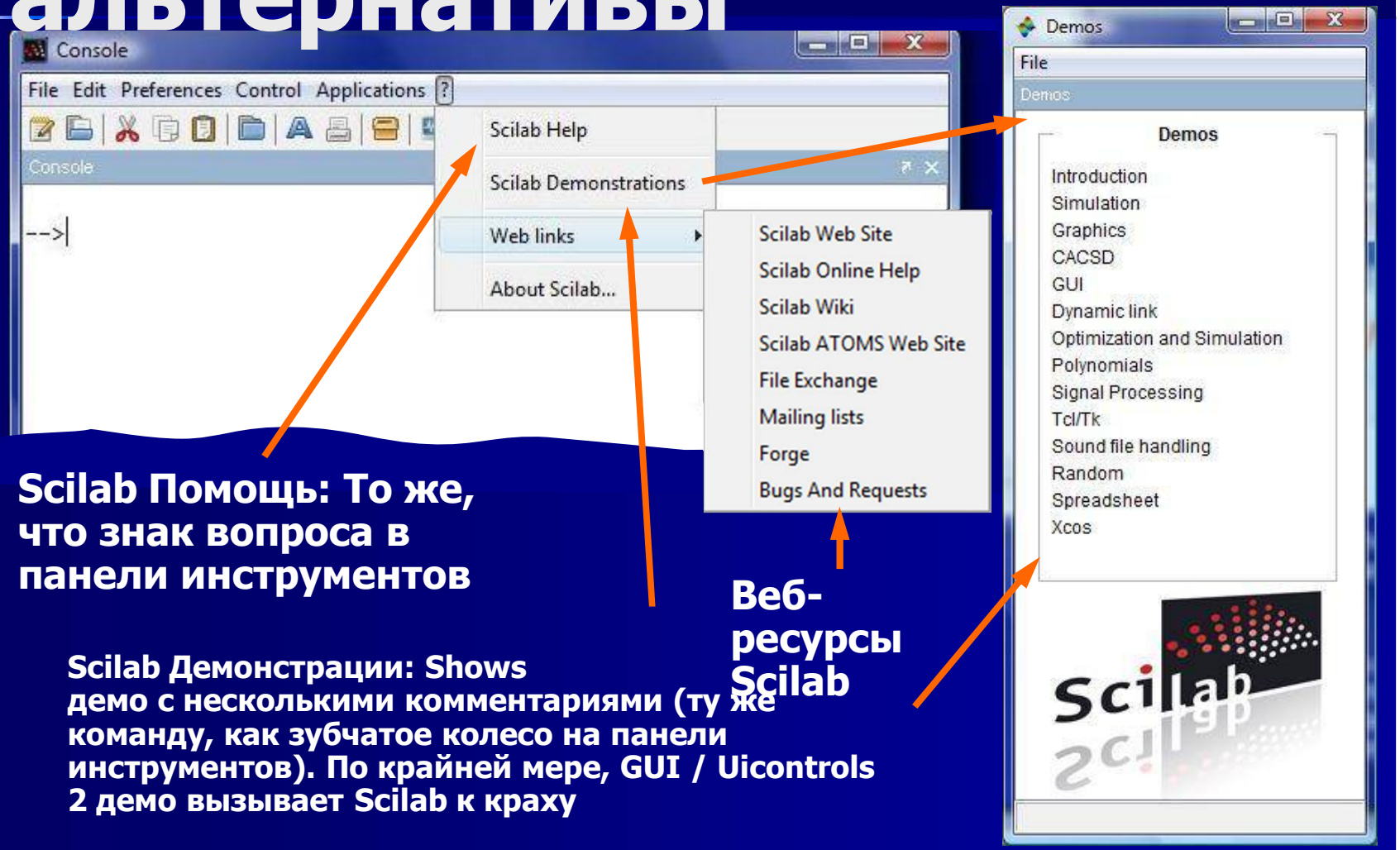
Атомы: Открывает онлайн менеджер модуль

Переменная Браузер: Открывает список с переменными (то же самое как команда browsevar)



История команд: Открывает список с используемыми командами.

Строка меню консоли(6/6): Помощь альтернативы



Scilab Помощь: То же, что знак вопроса в панели инструментов

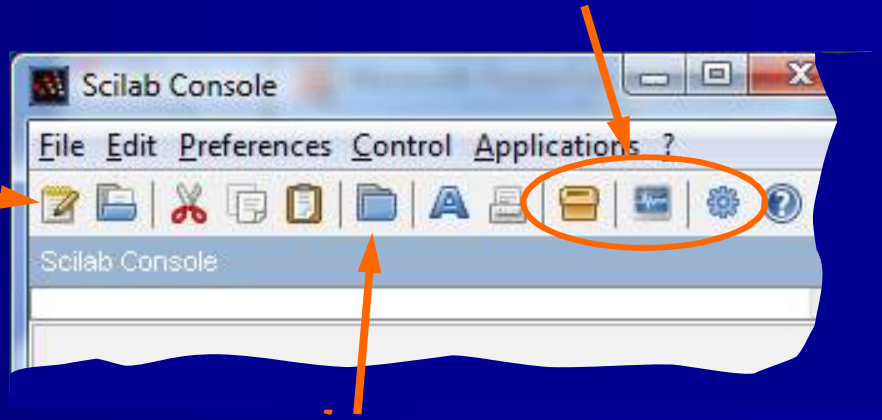
Веб-ресурсы Scilab

Scilab Демонстрации: Shows демо с несколькими комментариями (ту же команду, как зубчатое колесо на панели инструментов). По крайней мере, GUI / Uicontrols 2 демо вызывает Scilab к краху

Панель инструментов консоли

Атомы, Xcos, и демонстрации иконки пришли в Scilab 5.2

Запустите редактор:. Открывает редактор Scilab в (SciNotes, другую часть своей интегрированной среды разработки (IDE) Основные учебники редко обращают внимание на то, что обычно мы работаем в (писать, редактировать, сохранять, управлять) редакторе, а не в окне консоли. Редактор представлен на несколько слайдов ниже.



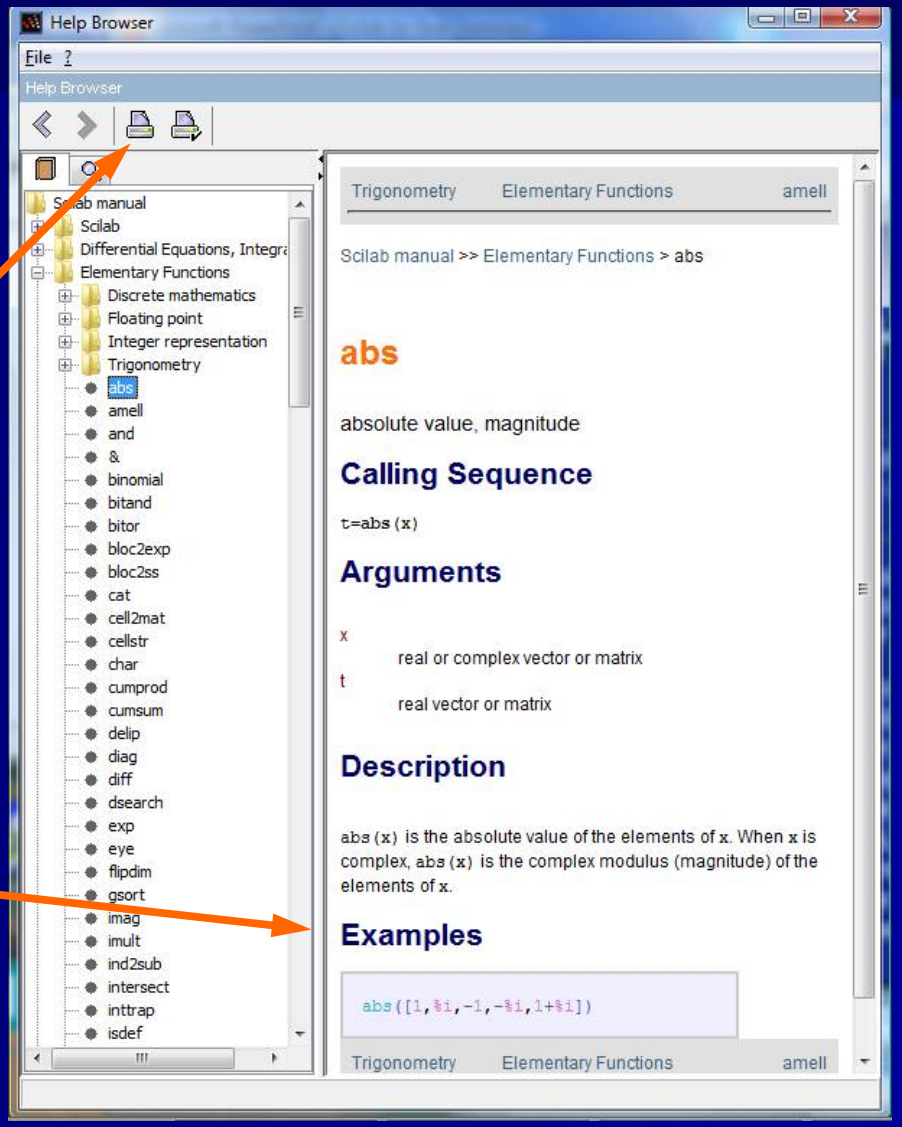
Изменение текущего каталога: Это также можно найти в файле в строке меню. Вам нужно отметить, с какой директории (папки) Scilab должен искать сценарий, который вы хотите выполнить (бег)

Помощь Браузер(1/3)

В консоли Нажмите на иконку Помощь браузера, чтобы открыть его

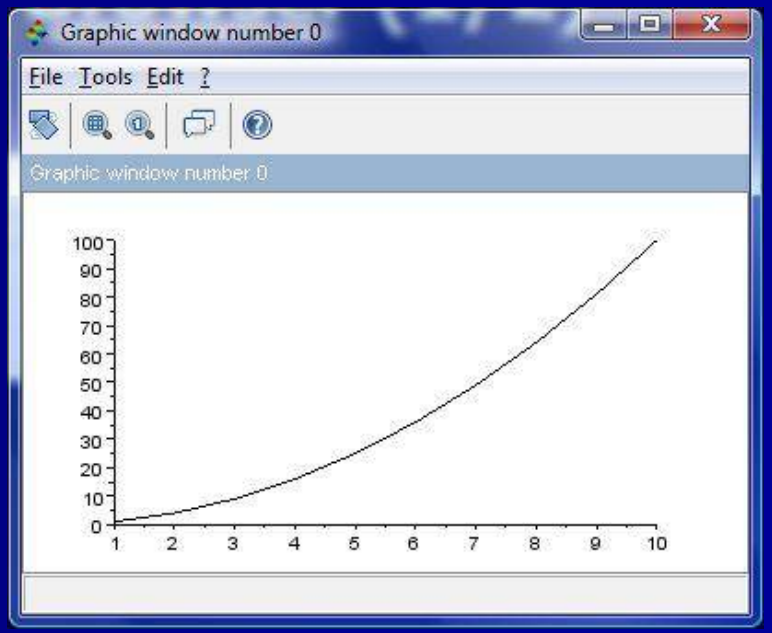
Помощь дискуссии становятся более читаемым, если распечатать их как PDF файлы.

Помощь Браузер является краткой "энциклопедией" основных возможностей и функций в Scilab. Объяснения функций дополнены примерами (см. следующий слайд для демо), но это не компенсирует хороший учебник.

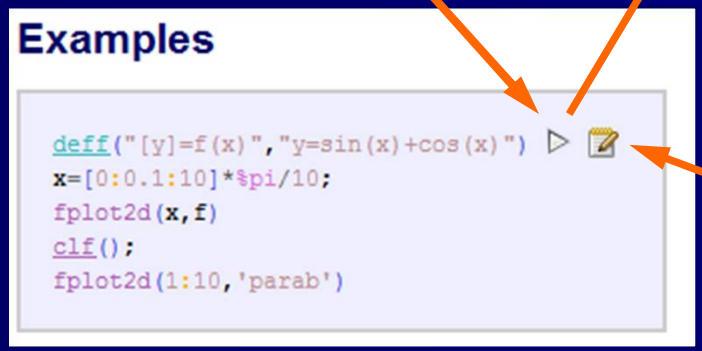


Помощь Браузер(2/3)

2. В Графическом окне с сюжетом всплывает (в этих случаях он кратко мигает первый участок)



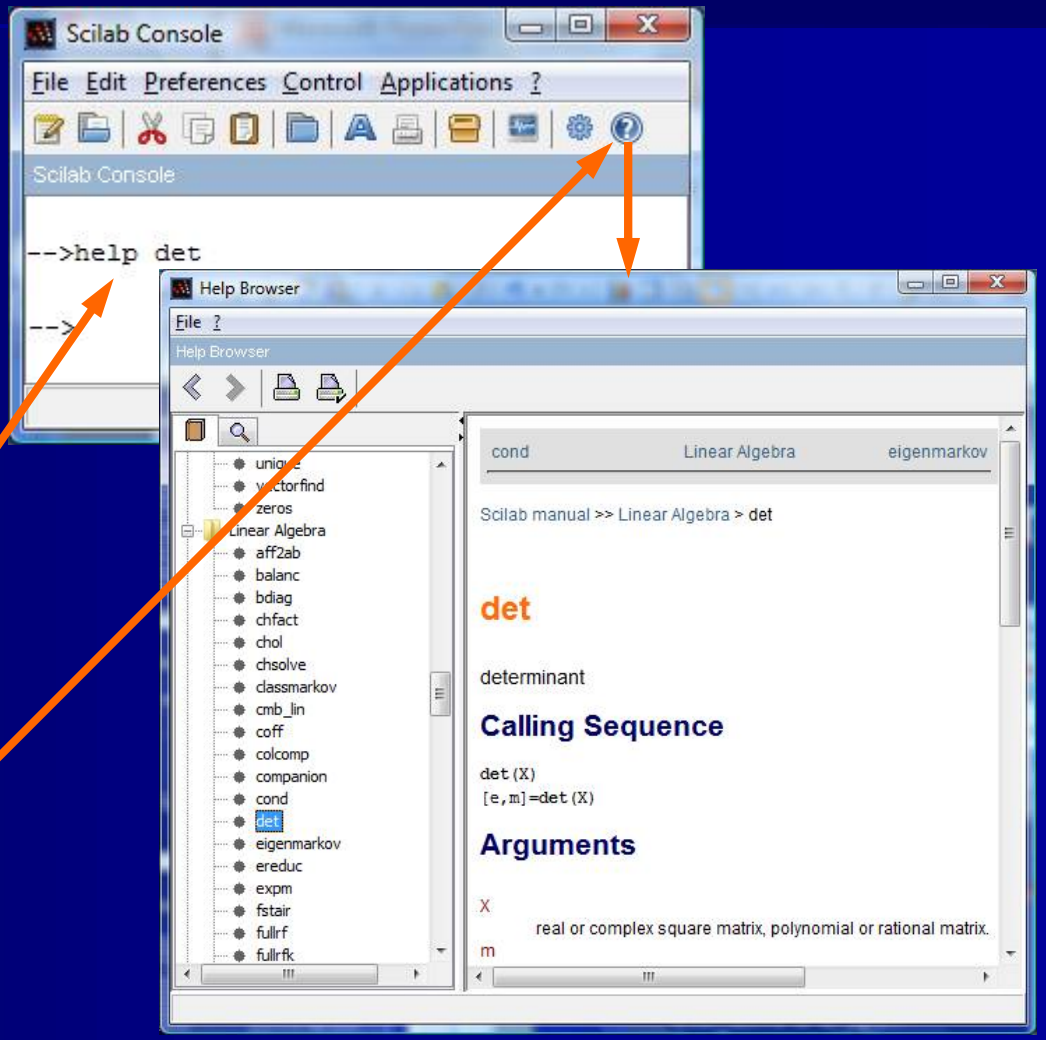
1. В скрипт коробке с примерами, кликните по значку Выполнить, чтобы увидеть, как скрипты Выполняет (не все работы)



3. Нажмите на иконку редактора и сценарий передается в текстовый редактор Scilab, где вы можете «играть» с ним (должны быть сохранены до того, как может работать)

Помощь Браузер(3/3): помощь имя_функции

Чтобы найти правильное использование любая функция-при условии, что имя известно- Помощь Браузер можно открыть из консоли, набрав команду командной помощь function_name, в показанном случае помощи опр () (скобки можно опустить). В качестве альтернативы можно открыть браузер со значком Помощь

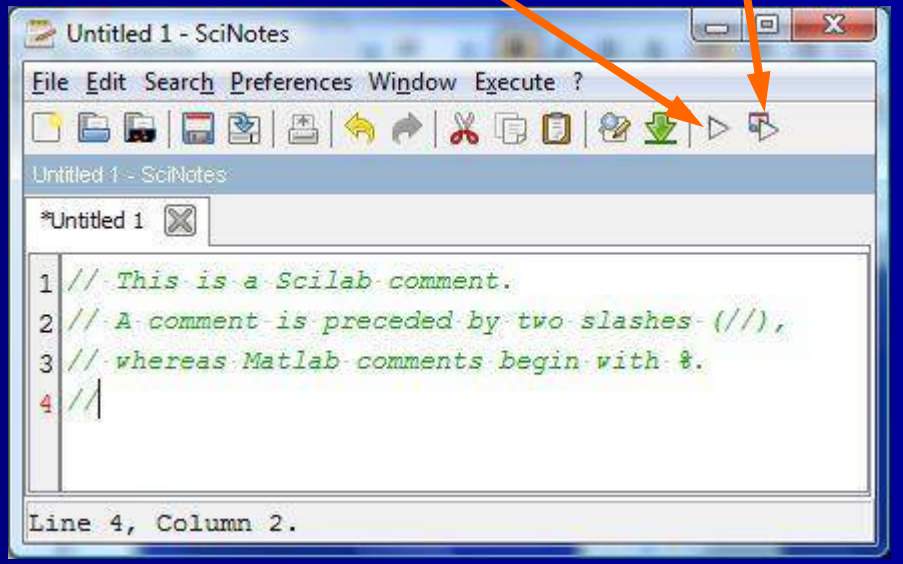


Редактор (SciNotes)

- Текстовый редактор где написаны исполняемые сценарии Scilab, поддерживаются и запустить .
- Откройте редактор, нажав на иконку запуска SciNotes в консоли, или нажав: Приложения \ SciNotes.
- Сценарий Scilab представляет собой текстовый файл с именем типа *. SCE
- Это хорошая практика, чтобы использовать сценарии для небольших задач. Тогда все "проекты" сохраняются и комментируются, готовы к использованию.

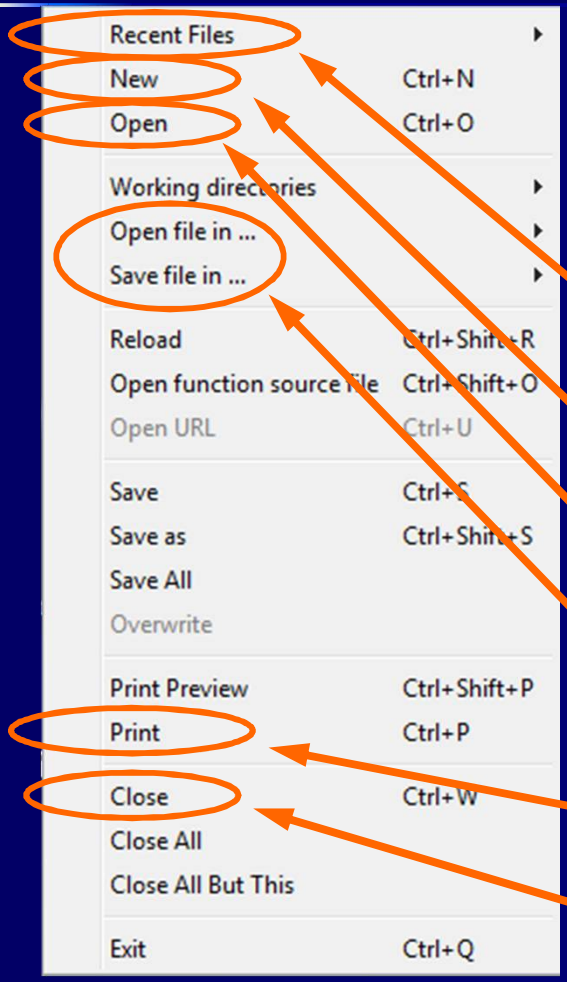
Сохранить и выполнить

выполнять



Но не забывайте, о правильном создании организованного архива ваших программ!

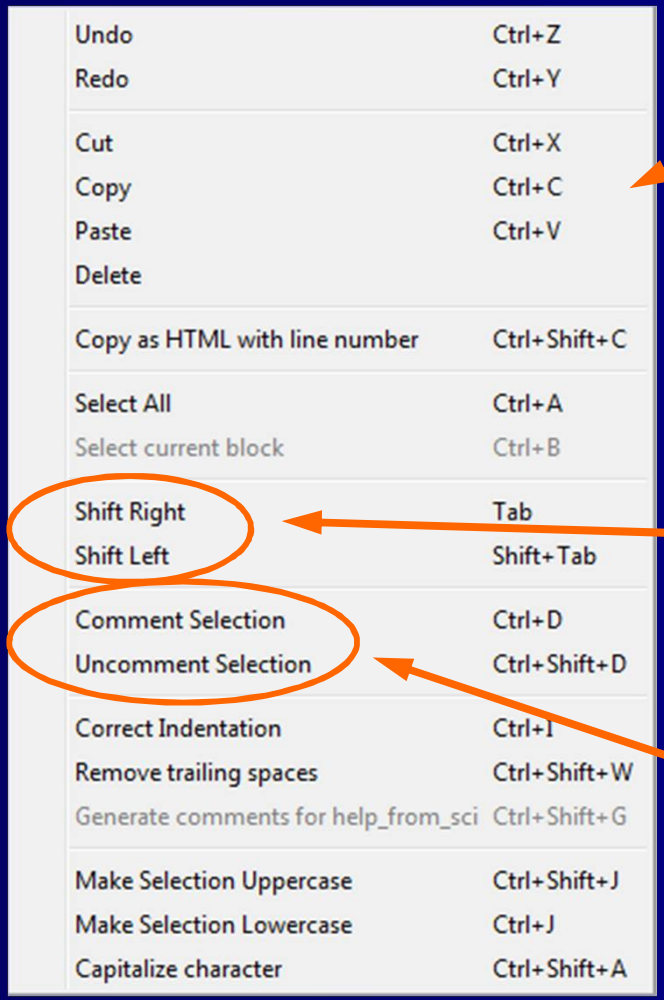
Строка меню Редактор (1/5): Файл

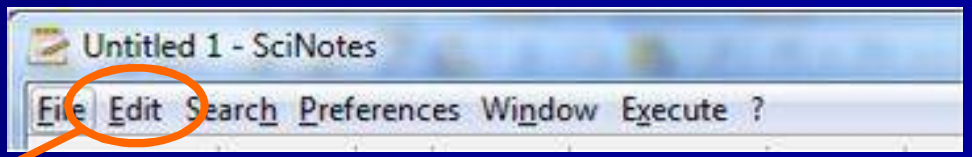


Команды файла, что вы, скорее всего, столкнетесь с:

- **Последние файлы дает быстрый доступ к недавно редактируемого текста**
- **Новый открывает вторую вкладку для нового сценария или редактирования**
- **Открыть открывает сохраненный сценарий в редакторе**
- **Открыть файл и Сохранить файл .. не работают в Scilab 5.3**
- **Печать является обычным команда печати**
- **Закрыть закрывает файл**

Строка меню Редактор (2/5): Редактировать



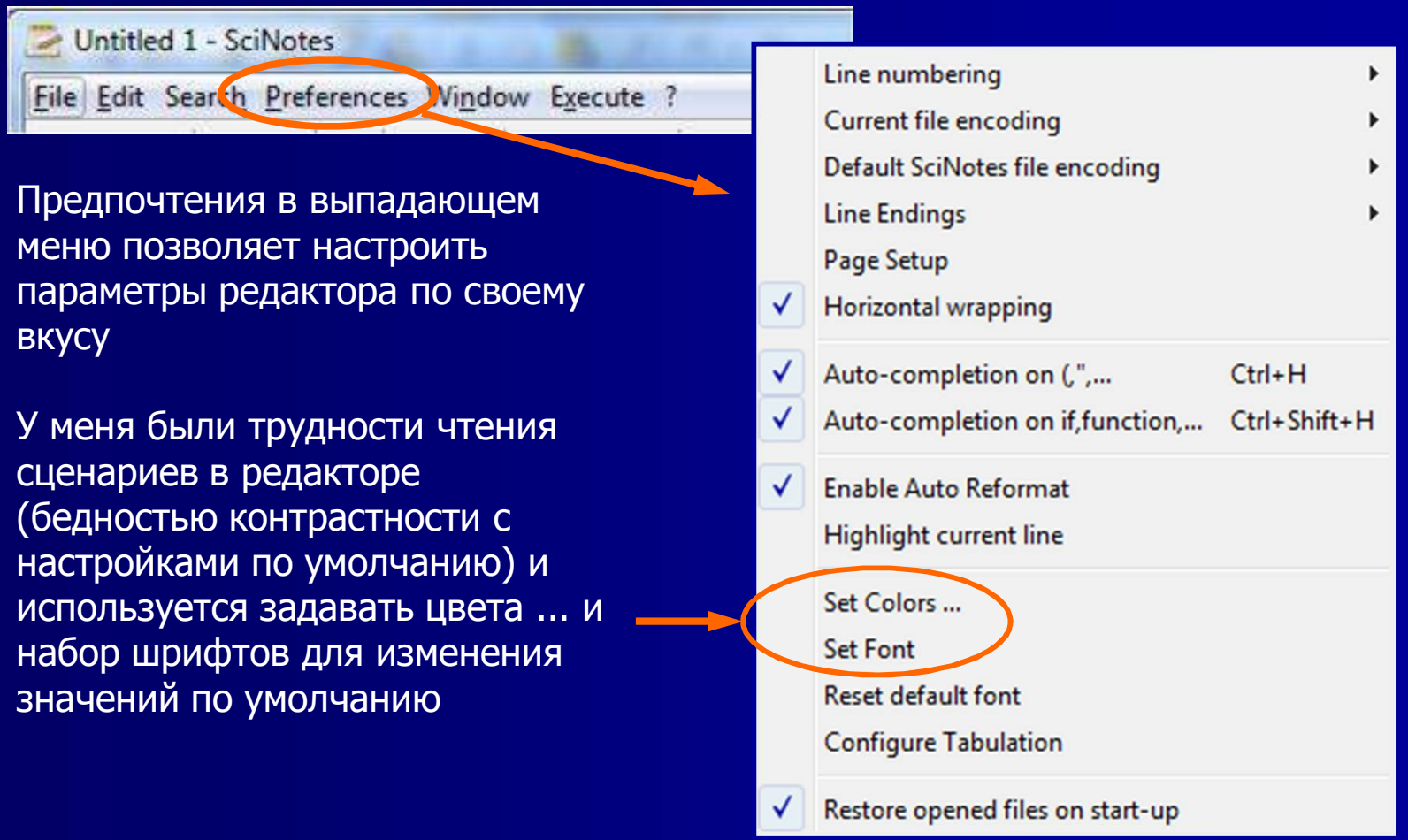


Команды под Edit в основном сами по себе. Однако обратите внимание на следующие четыре:

Сдвиг вправо / влево: отступ / Unindent строку на один шаг (эта пара должна быть на панели инструментов)

Комментарий / раскомментировать Выбор: Добавить / удалить комментарий

Строка меню Редактор (3/5): Предпочтения



Предпочтения в выпадающем меню позволяет настроить параметры редактора по своему вкусу

У меня были трудности чтения сценариев в редакторе (бедностью контрастности с настройками по умолчанию) и используется задавать цвета ... и набор шрифтов для изменения значений по умолчанию

Строка меню Редактор (4/5): Предпочтения, комментарий

Пользователи могут отправлять отчеты об ошибках исследовательской команде Scilab в (ссылка на <www.scilab.org>). Я подал следующий доклад (Bug 8802):

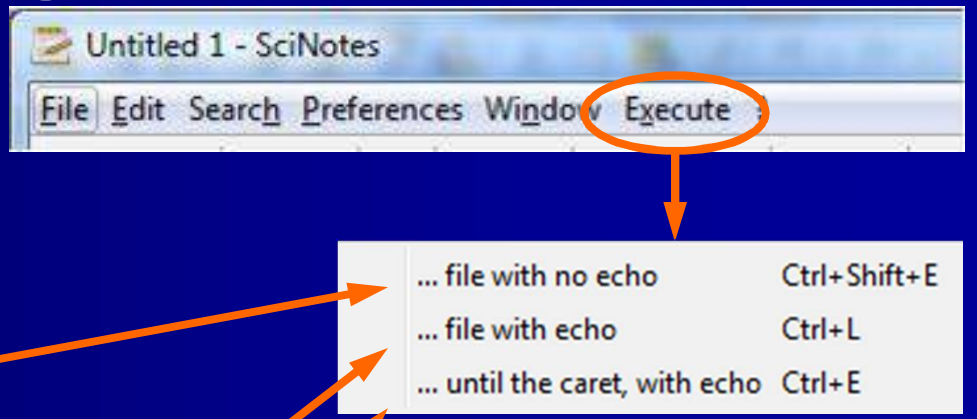
«По умолчанию настройки цвета в редакторе неконтрастен ... Изменение цвета шрифта утомительно из-за чрезмерного количества вариантов под Предпочтения \ установить цвета ... (излишним, можно сказать). Я хотел бы предложить настройки по умолчанию только с четырьмя цветами (красный, зеленый, синий и черный).»

На что я получил ответ:

«Вы можете проще изменить конфигурацию цвета в изменении файла: C: \ Documents и Settings \ Джонни \ Application Data \ Scilab \ Scilab-5.3 \ scinotesConfiguration.xml (или путь, который похож)»

Я нашел scinotesConfiguration.xml под C: \ Program Files \ Scilab-5.3 \ модулей \ scinotes \ и т.д. \. Цветовые коды XML должен быть изменен в этом файле. Я желаю вам удачи

Строка меню Редактор (5/5): Выполнить



Выполнить раскрывающемся окно содержит три варианта:

... Подать, не эхо: простое выполнение команды (то же, нажав на значок Выполнить на панели инструментов)

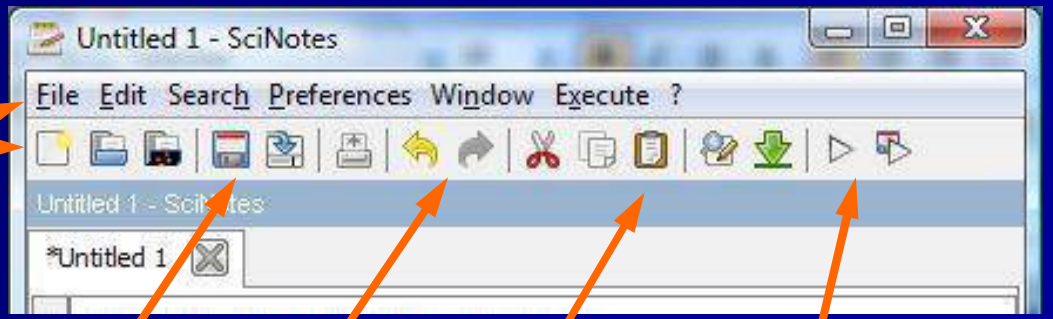
... Подать с эхом: Выполняет сценарий и перекликается на него (показывает его): на консоли

... Пока каретка, с эхом: это выше моего понятия

Выполнить команды, используемые «файл» проще. Я понятия не имею, почему они изменили их таким образом. Моя рекомендация заключается в использовании значок Выполнить на панели инструментов (см. следующий слайд)

Панель инструментов редактора

Новый ... Открывает вторую вкладку для нового сценария для редактирования (та же команда может быть найден в файле)



Сохранить иконка выглядит как голландской триколор, но вы привыкнете к нему. Следующий из них Сохранить как ...

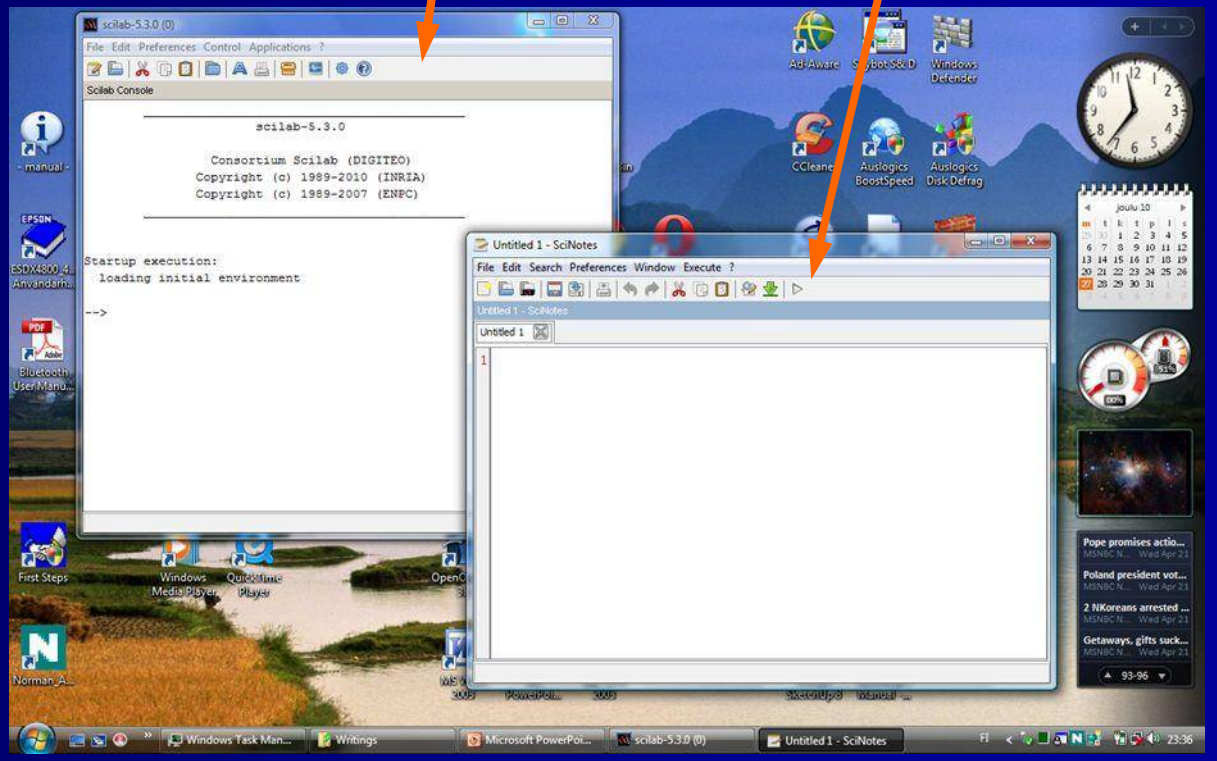
Стрелки отмена / повтор вполне привычны

Значок Вставить

Выполнить (или Сохранить и выполнить) значок то, что вы обычно используете для запуска сценария

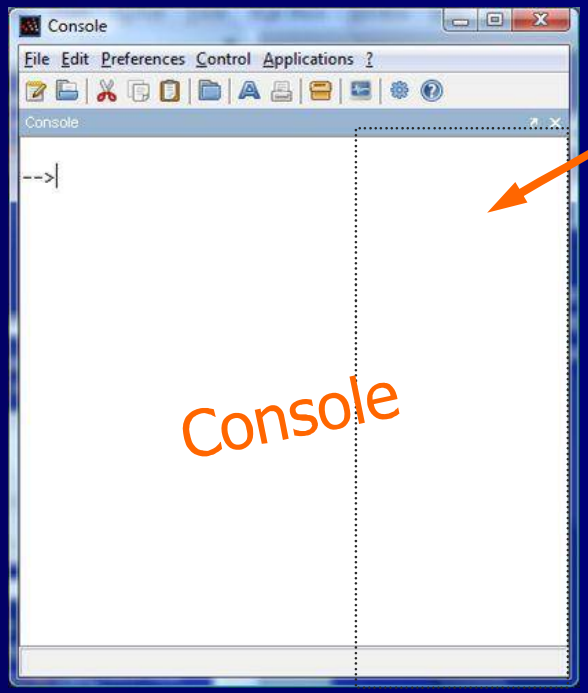
Готовность к работе

Ваш рабочий стол должен выглядеть примерно так, как здесь. Как мы видели, редактор и консоль необходимы, так как при сценарии, созданных в редакторе выполняются в консоли.

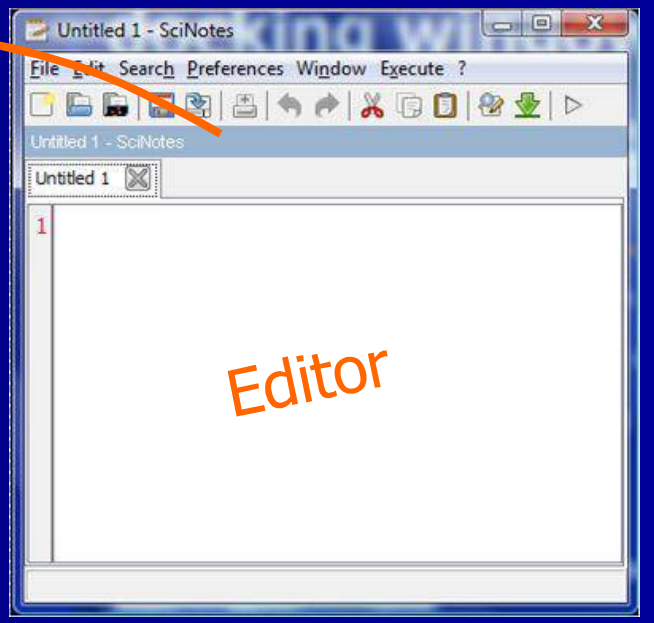


И еще один момент (1/2): стыковки окон

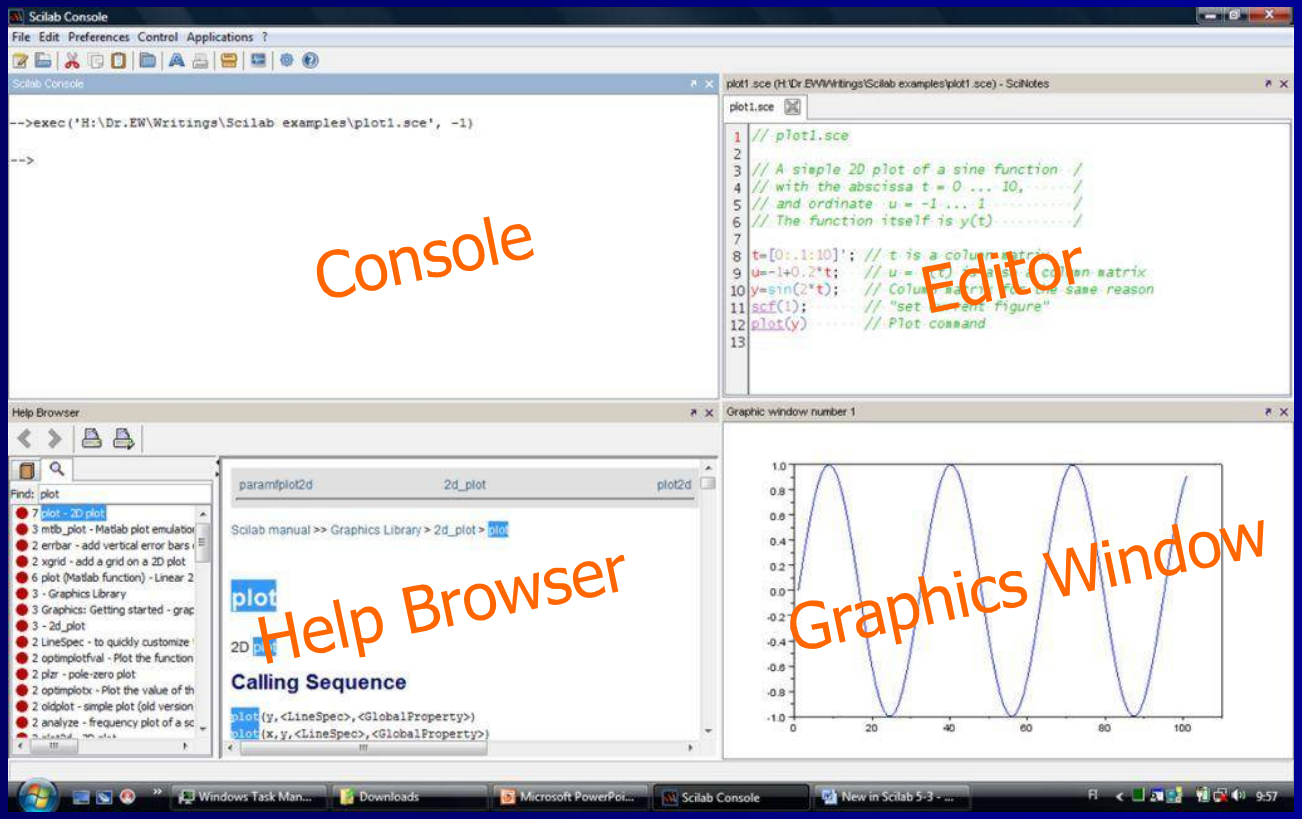
- **Можно прикреплять окна Scilab, т.е. сформировать единую рабочую область похожую на ту, что в Matlab. Вот как это сделать:**



Нажмите левую кнопку мыши на затемненной строке активного окна, перетащите в другое окно и отпустите. На следующей странице приведены варианты.



И еще один момент (2/2): стыковки окон



Каждое окно часть имеет стрелку в правом верхнем углу, с помощью которого можно освободить ее от стыковки

Сценарии и функции

- Scilab имеет два типа команд:
  Сценарии. Набор команд используется для автоматизации вычислений. Команды сценария, как правило, возвращается к консоли, но участки возвращаются в окне с графикой
  Функции (макросы). Короткие программы, взаимодействие с окружающей средой через входных и выходных переменных.
- Список наиболее часто используемых встроенных функций дается на следующем слайде. Функции, определенные пользователем могут быть локальными (интегрированные в сценарии) или глобальными (хранятся в отдельном файле и доступны для любого сценария)
  Я могу использовать термин «код», чтобы сделать общие ссылки на либо сценарии или функций
  Как уже было сказано и будет повторяться-то: следовало бы скорее создавать сценарии и функции на (Text) Editor (SciNotes)

Встроенные функции

Ниже приведен список наиболее распространенных математических функций в Scilab. Полный список встроенных функций можно найти в разделе Помощь \ элементарных функций, которые также объясняет требования к аргументам (Есть и обязательные и необязательные аргументы).

sin(), cos(), tan(), cotg()	Тригонометрические функции, e.g. sin(.2*%pi)
asin(), acos(), atan()	функции Arc
sinh(), cosh(), tanh(), coth()	Гиперболические функции
asinh(), acosh(), atanh()	Обратные гиперболические функции
sqrt(), exp()	Корень квадратный, например SQRT (2) / показатель
sum()	Сумма
min(), max()	Минимальное / максимальное значение
abs(), sign()	Абсолютное значение, например, ABS (синк (x)) / знак
real(f), imag(f)	Реальные и мнимые части комплексной функции

Предопределенные переменные и константы

Основные предопределенные, защищенные от записи переменных / константы:

%i	$i = \sqrt{-1}$	Мнимая единица
%pi	$\pi = 3.1415927\ldots.$	Число пи
%e	$e = 2.7182818\ldots.$	Napier's constant e
%eps	$\varepsilon = 2.22 \cdot 10^{-16}$	Точность (машиннозависимый)
%inf		Бесконечные (не математически бесконечное)
%nan		Не Количество
%s	s	Полином переменной
%z	z	Полином переменной
%t, %T	true	Логическая переменная
%f, %F	false	Логическая переменная

Операторы Scilab (1/2)

Список содержит большинство операторов, используемых в Scilab. Многие из них будут подробно описаны позже.

;	Конец выражения, разделение строки
,	Инструкция, аргумент или столбец
'	Конъюгат (матрица) транспонирования, строка разделитель *
.'	Номера для сопряженной транспозиции
[] , [] '	Вектор или матрица определенное объединение, транспонированная матрица
()	Пара левой / правой скобках используется для различных целей
+ , -	Сложение, вычитание
* , .*	Умножение, элемент – на - элемента

***) Как простой (') и двойные (") кавычки позволили определить символьные строки**

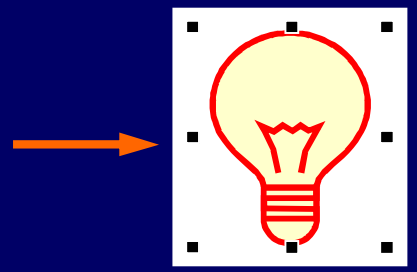
Операторы Scilab (2/2)

/ , ./	Правое деление, элемент-на-элемента деление справа
\ , .\	Левое деление, элемент -на элемент
^ or ** , .^	Мощность (показатель), элемент-на-элемента ия
.*.	Кронекера продукт
./. , .\.	Кронекера направо и налево разделение
|	Логическое ИЛИ
&	Логическое И
~	Логическое НЕ
==, >=, <=, >, <, <>, ~=	Равно, равной или больше, равно или меньше, чем, больше чем, меньше чем, не равно (двух альтернатив)

Вычислительная терминология: краткое введение

- **Аргументы**: Ценности, предусмотренные в качестве вклада в команде (входных аргументов) или возвращаемых командой (выходными аргументами)
  Команда: Пользовательское письменное заявление, которое содержит инструкции к компьютеру ("заявление" является часто используется альтернативный)
  По умолчанию: Меры или значения, выбранные программой, если такие не были предоставлена
  Дисплей: Вывод текстовой информации на экран компьютера
  Эхо: Для отображения команд или другой вход, вводимые пользователем
  Выполнить: Чтобы запустить программу или выполнять инструкции, указанные в команде
  Печать: Для вывода информации компьютерным принтером (часто путают с "дисплей")
  Возвращает: Результаты, предоставляемые компьютером в ответ на команду

На "ручками"

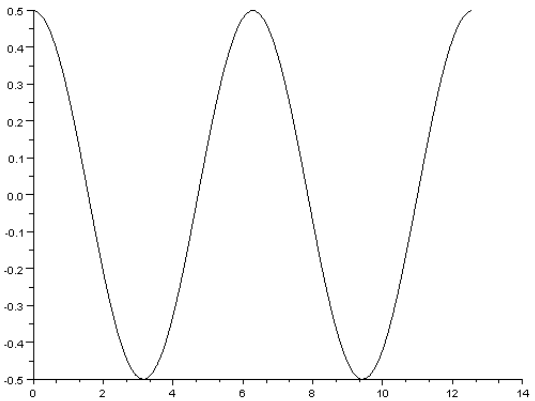


Вы будете часто видеть Scilab в Помощь Браузер относятся к "ручке", но Помощь не обеспечивает полезное объяснение этого термина. Вот краткое изложение:

- В графическим программам слово «ручка» слово относится к точкам, расположенных вокруг фигуры или сценария, которые позволяют манипулировать объектом? (См. рисунок)
  Учебник Matlab дает следующее объяснение, которое справедливо и для Scilab: "Всякий раз, когда Matlab создает графический объект, он присваивает идентификатор (так называемый дескриптор) к нему. Вы можете использовать эту ручку для доступа к свойствам объекта ".
  Вам нужно ручки редактирование графических участков не по карману, предлагаемых основных функций участок (plot2d (), plot3d () и т.д.)
  Мы вернемся «ручки» при обсуждении графика.

Проверьте ручки? С GCF ()

- Функция plot2d () производит следующую участок
  Команда GCF () дает список вправо
  Список является ручка для определенной функции (Scilab литературе также относится к отдельным строкам в списке под термином «ручкой»)




```
-> x = LINSPACE (0,4 *% пи, 100); plot2d (x,
0,5 * соз (x))

-> F = GCF ()

e =

Ручка типа "Рисунок" со свойствами:
==========================
=========
детей: "Топоры"
figure_position = [567485]
figure_size = [628592]
axes_size = [610460]
auto_resize = "на"
видового = [0,0]
figure_name = "Графический номер окна% г"
figure_id = 0
info_message = ""
color_map = матрица 32x3
растровое изображение = "выключено"
pixel_drawing_mode = "копировать"
anti_aliasing = "выключено"
immediate_drawing = "на"
фон = -2
видно = "на"
rotation_style = "унарный"
-обработчик = ""
event_handler_enable = "выключено"
user_data = []
тег = ""
```

Foo

- Термин "Foo" используется во многих учебников. Это может ввести в заблуждение, если вы не тесно знакомы с программированием .
  Проще говоря, Foo можно интерпретировать как "что-то приходит сюда." Профессиональный выражение имя заполнитель, также упоминается как метасинтаксическая переменной
  Пример:

```
for k = 1:2:n
   foo;
end
```

- Альтернативные названия заполнители, которые вы можете встретить в Foobar, бар, и Баз. Я предпочитаю использовать точки (....)

Dr. EW
Johnny Heikell

3. Игра с консолью и редактором

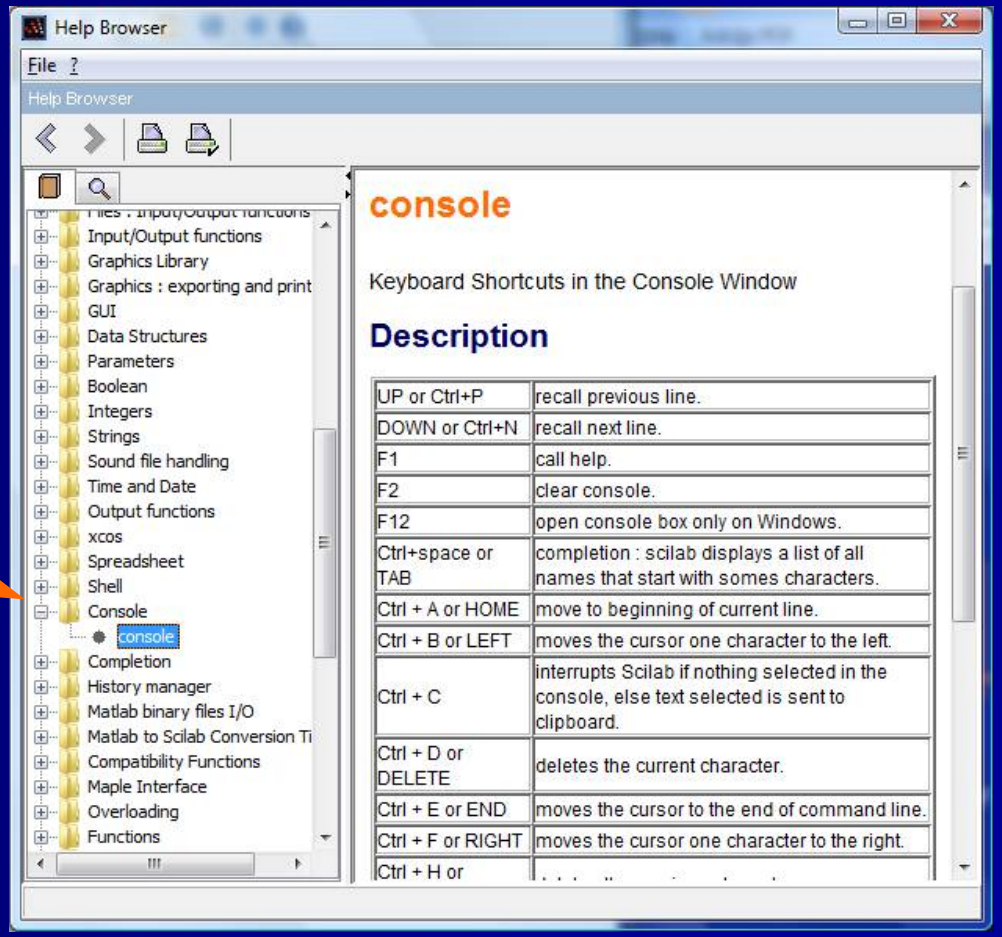
Эти неуклюжие первые шаги;
немного о том, что Scilab делает

Сочетания клавиш консоли

Сочетания клавиш позволяют быстрее выполнять команды, но требуют частого использования остается запомнить

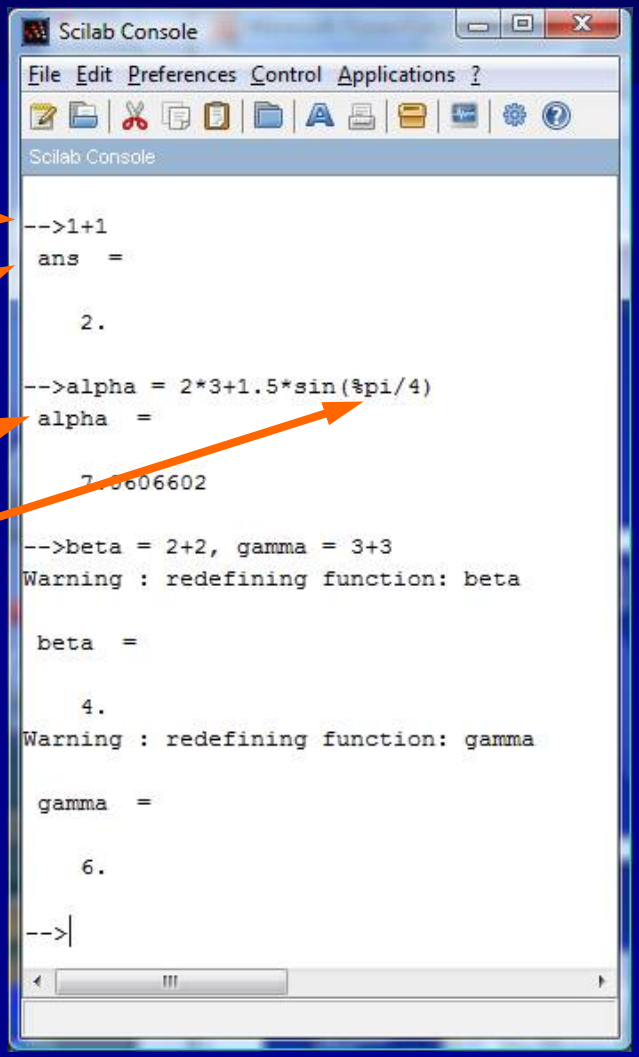
В Помощь браузера, выберите пункт: Console / консоли для получения списка сочетаний клавиш

Простейшие из них для запоминания являются:
F1 = Открыть Помощь Браузер
F2 = Очистить консоль



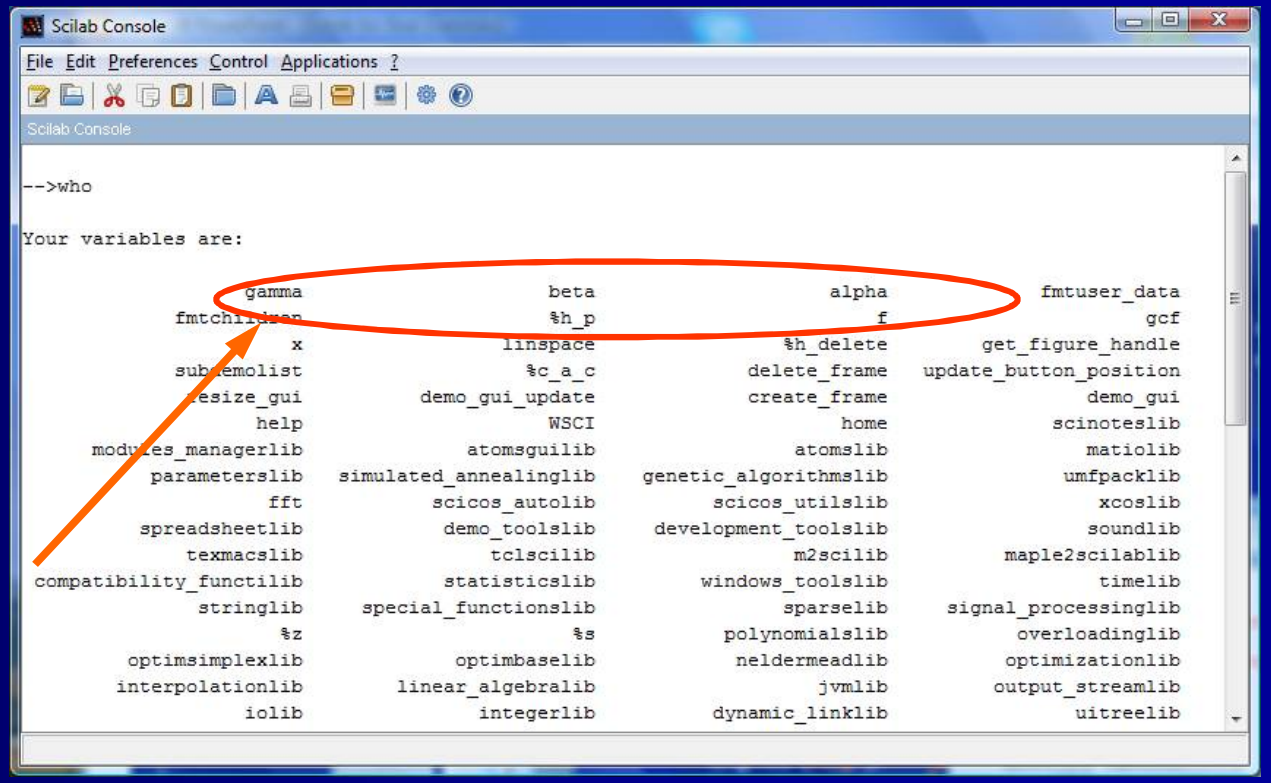
Простые расчеты

- Консоль может использоваться как калькулятор, написав арифметические выражения после командной строке и нажав Enter
  Если ни одна переменная не задана, Scilab использует встроенную переменную ans.
- Когда переменная задается (здесь alphs) оно будет использоваться вместо этого. **pi** является встроенной переменной (постоянной) %pi.
- Выражения можно записать в одной строке, разделяя их запятой (предупреждение может быть проигнорировано) .
- Scilab отображает выполненную команду, если она не заканчивается точкой с запятой (;)



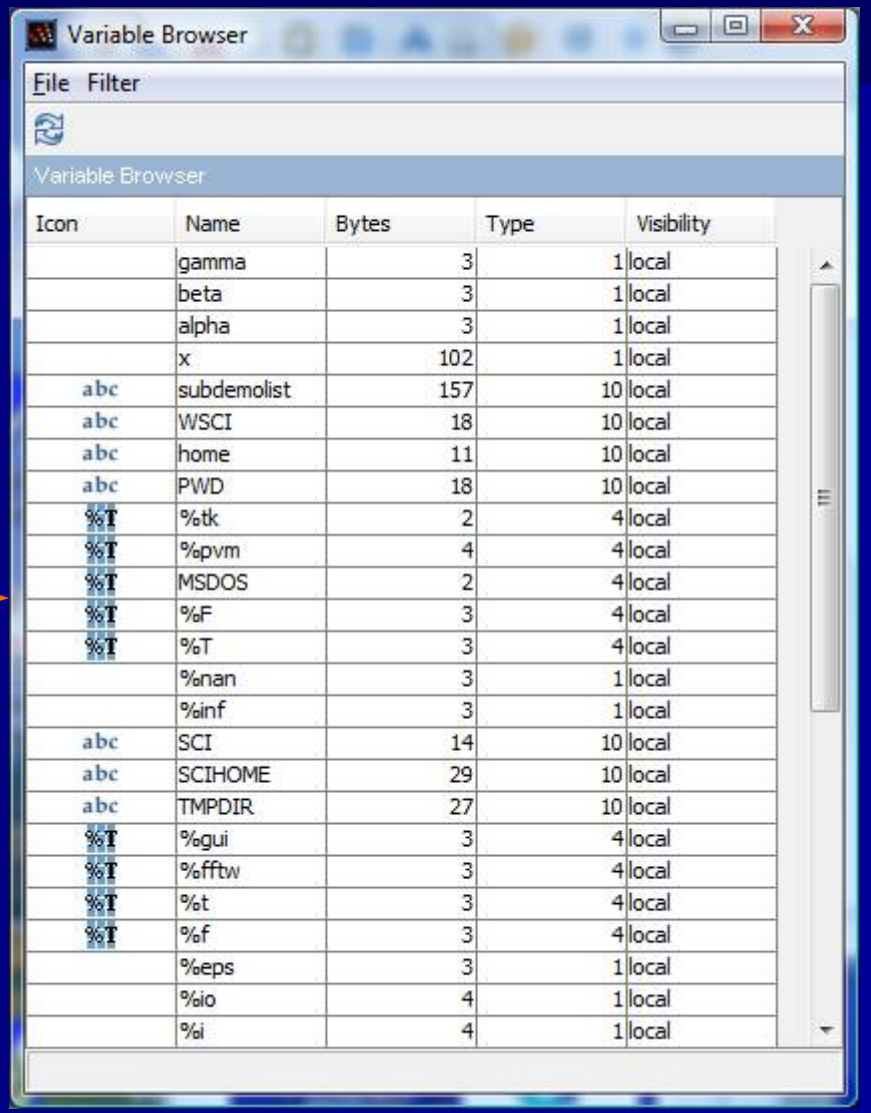
Список переменных (1/2)

Команда, **who** (+ Enter) создает список некоторых переменных Scilab. По крайней мере, на моем ноутбуке Windows Vista столбцы по правому краю . Обратите внимание, что переменные из предыдущего примера отображаются.



Список переменных (2/2)

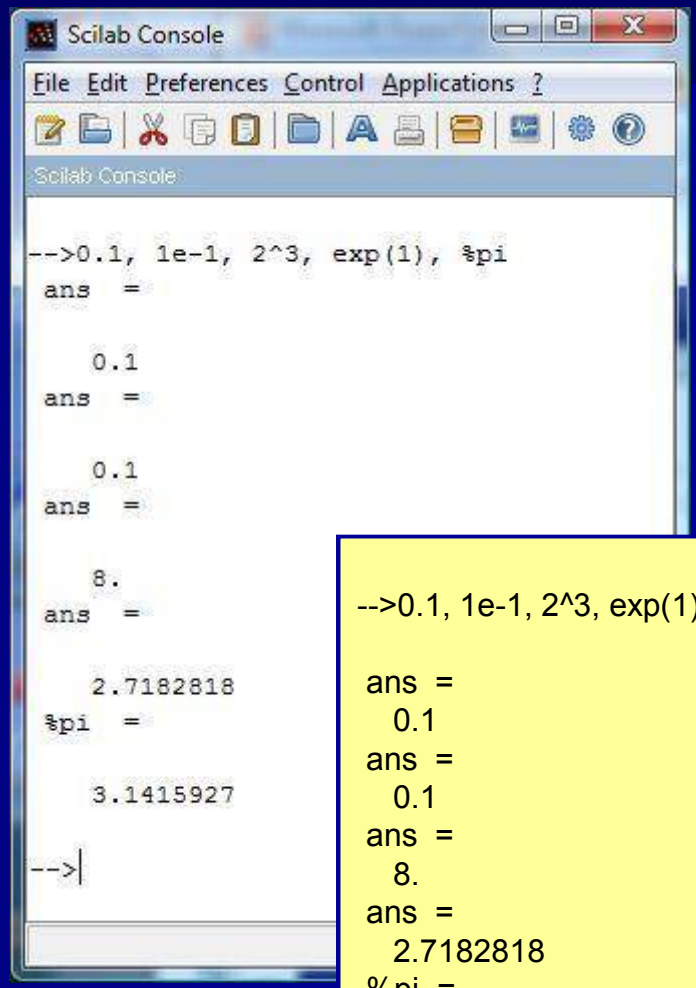
- Команда **browsevar** открывает Variable окно браузера .
- Список, который всплывает дает информацию о типе и размере каждой переменной.
- Напомним, что переменная Браузер также можно назвать через меню: Приложения / Переменная браузера



Icon	Name	Bytes	Type	Visibility
	gamma	3	1	local
	beta	3	1	local
	alpha	3	1	local
	x	102	1	local
abc	subdemolist	157	10	local
abc	WSCI	18	10	local
abc	home	11	10	local
abc	PWD	18	10	local
%T	%tk	2	4	local
%T	%pvm	4	4	local
%T	MSDOS	2	4	local
%T	%F	3	4	local
%T	%T	3	4	local
	%nan	3	1	local
	%inf	3	1	local
abc	SCI	14	10	local
abc	SCIHOME	29	10	local
abc	TMPDIR	27	10	local
%T	%gui	3	4	local
%T	%fftw	3	4	local
%T	%t	3	4	local
%T	%f	3	4	local
	%eps	3	1	local
	%io	4	1	local
	%i	4	1	local

Ввод номеров

- В Scilab цифры могут быть введены по-разному, как показано в этом примере.
- Некоторые выражения имеют альтернативные формы. Например, есть три силовые выражения (^), (**) и (. ^), Но Scilab берет их в этом порядок вызова .
  Обратите внимание, что ans и pi предоставлены с семью знаками после запятой, что ставит пределы достижимой точности (существует функция двойной точности)
- Обратитесь за помощью, если вам нужно изменить формат отображения
- Примечание: Отныне я показываю только содержимое консоли (на светло-желтом фоне).



Точность вычисления (1/2)

```
-->a = 1 - 5*0.2

 a  =
    0.


->b = 1 - .2 - .2 - .2 - .2 - .2
 b  =
    5.551D-17
```

Посмотрите на эти два примера слева. В обоих случаях мы вычисляли 1-5 * 0,2, но двумя различными способами.

В первом случае ответ правильный (0) .

Во втором случае ответ 5,55 * 10-17, которые совершенно очевидно неверен.

Причина в том, что ответ имеет конечную точность (ошибок округления) .

Мы должны принять это ограничение во внимание, в процессе моделирования Scilab.

Точность вычислений (2/2)

Вот еще два случая, когда проявляется конечная точность. Ответы должны быть 0 (ноль) и T (True) соответственно (Обратите внимание, что 1.225D-15, 1.225e-16, 1,225 * 10 ^ -16 и 1,225 * 10-16 то же самое).

```
-->a = sin(%pi)

 a  =


    1.225D-16
```

Предположим, что указанная переменная является частью сценария с id ..., then ... else... . Результатом является то, что вариант 1 никогда не выполняется, потому что никогда не может быть точно равен нулю.

```
foo
if a == 0 then
     alternative 1
  else
     alternative 2
end
```

Мы должны проверить с некоторыми конечных пределах, например:

if ans (a) <1e-6 then.....

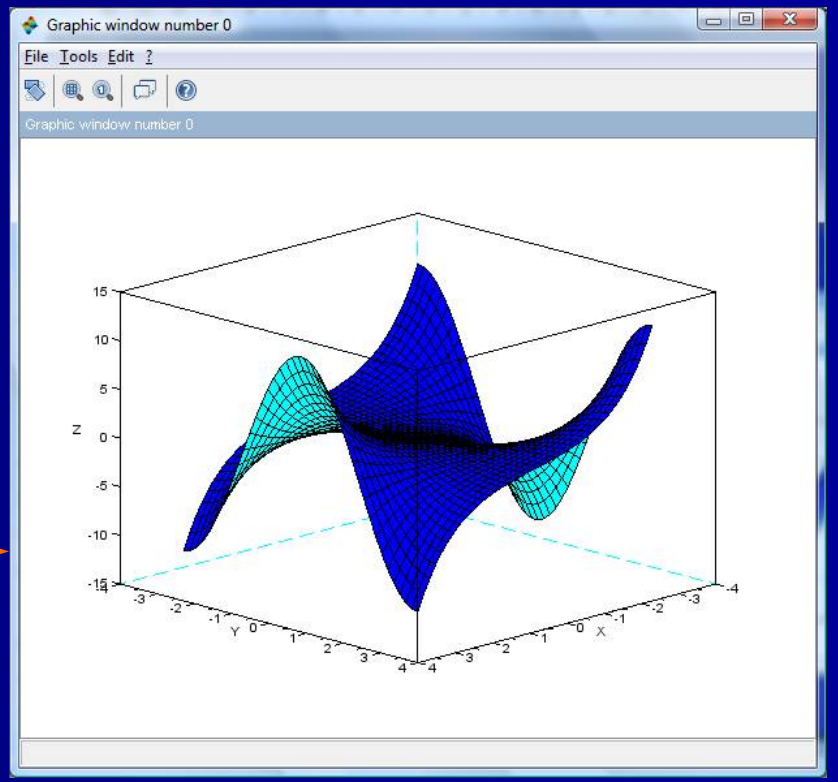
$|a| < 10^{-6}$

Изображение графика

- Консоль также может быть использована, чтобы дать команды для построения графика:

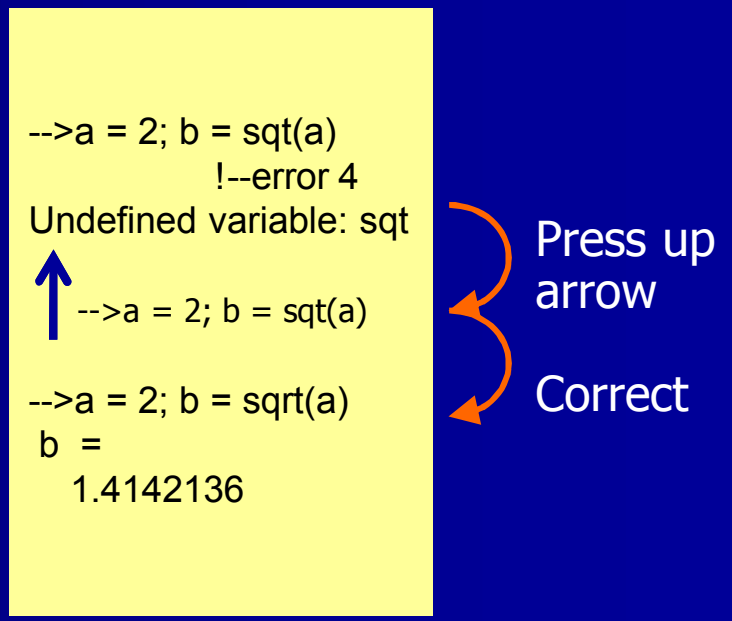
```
-->x = linspace(-%pi,%pi,40);
-->y = linspace(-%pi,%pi,40);
-->plot3d(x,y,sinh(x')*cos(y))
```

- Графический рисунок строится в графическом окне, которое появится автоматически.
- Смысл введенного кода станет ясным из дальнейшего.



Редактирование командной строки

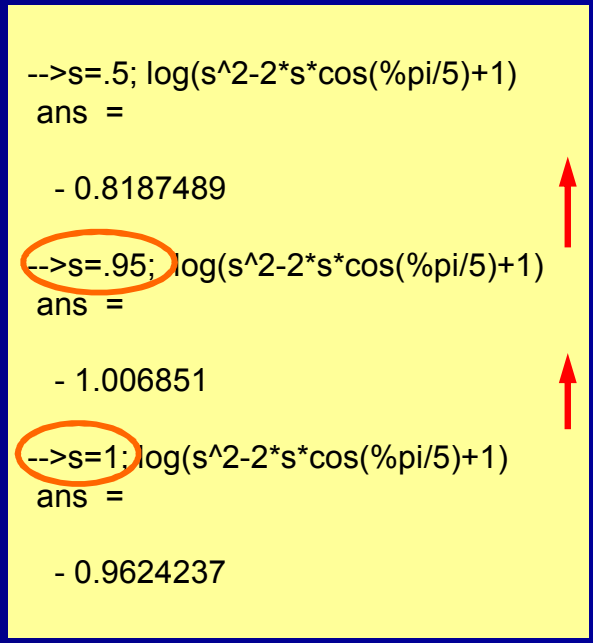
- Предположим, что мы допустили ошибку при вводе командной строки и Scilab выдает сообщение об ошибке.
- Вместо того, чтобы перепечатывать всю строку, мы можем нажать стрелку вверх () на клавиатуре, чтобы вернуть строку и исправить ошибку
  В показанном примере функция для квадратного корня, SQRT (), был первым ошибочно ввели SQT ().
- Отметим, что это лишь один из нескольких альтернатив для редактирования командной строки.



Редактирование демо

- Вычисление функции:

Log(s^2 - 2s *COS (%pi/5) + 1) для c = 0,5, 0,95 и 1?
Не переписывайте функцию, используйте вместо этого стрелки вверх, для редактирования предыдущей команды!



Комплексные числа

- Scilab обрабатывает комплексные числа же легко, как действительные
  Переменная %i расшифровывается как √ (-1).
- Первый пример показывает, как Scilab оценивает некоторые функции с комплексным аргументом x = 2 + 3i.
- Аргумент дает результат!?
- Второй пример показывает, как Scilab делает арифметические операции с два комплексных уравнений, x и y.

```
-->x = 2 + 3*%i;
-->abs(x)
 ans  =

   3.6055513
-->real(x)
 ans  =
   2.
-->imag(x)
 ans  =
   3.
-->sin(x)
 ans  =
   9.1544991 - 4.168907

-->atan( imag(x), real(x) )

 ans  =

   0.9827937
```

```
-->x = 2 + 3*%i; y = 1 - 1*%i;
-->z1 = x –y
 z1  =

    1. + 4.i

-->z2 = x * y
 z2  =

    5. + i

-->z3 = x / y
 z3  =

  - 0.5 + 2.5i
```

Векторные функции

- Функции Scilab являются векторными, это означает, что функции могут быть определены векторами.
- В показанном примере, создается вектор-столбец.
- Следующая вектор используется в качестве аргумента функции (), выраженного для y.
- Если значения t не представляют интереса, можно избежать вывода, поставив точку с запятой после выражения для t : t = [0:5] '; y = sin (0,2 * t).

```
-->t = [0:5]'
 t  =
    0.
    1.
    2.
    3.
    4.
    5.

-->y = sin(0.2*t)
 y  =
    0.
    0.1986693
    0.3894183
    0.5646425
    0.7173561
    0.8414710
```

Длинные ряды команд

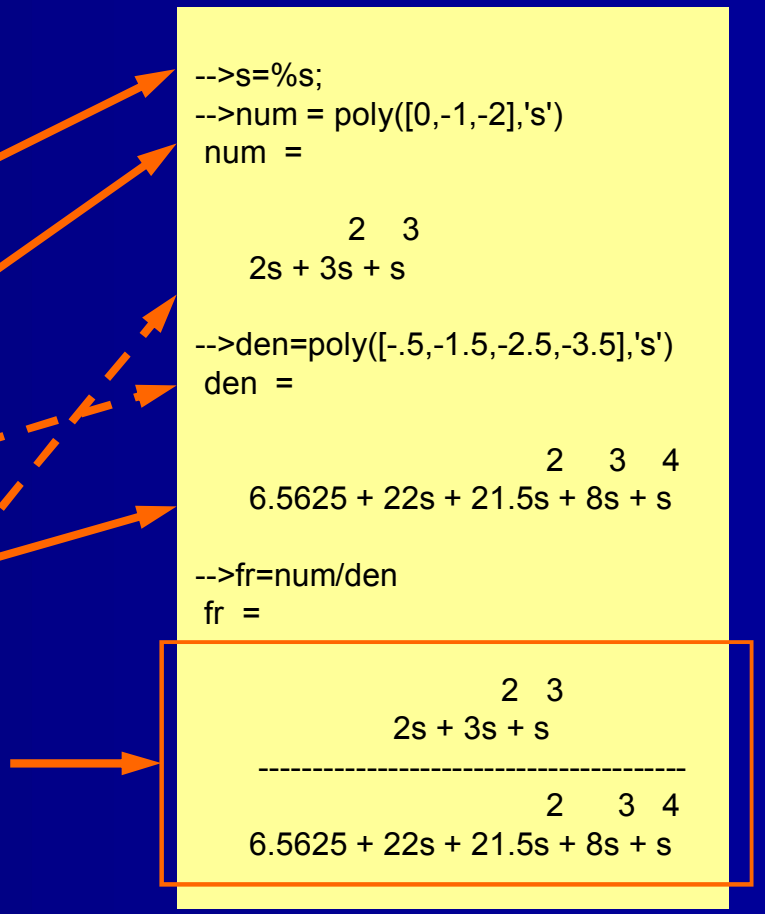
- Длинные выражения команд могут быть разделены между двумя и более линиями.
- Одним из инструментов для этой цели является два или три точки (..), чтобы указать, что запись продолжается.
- Длинные матричные выражения могут быть написаны на отдельных строках, опуская точку с запятой, которые обычно заканчивают ряд (нижний).

```
-->p=1+2+3+4+5+6+7+8+9+10+11+12+...
-->13+14+15+16+17+18+18+19+21+22+23+24+25
 p  =

    323.

-->q = 1/2 + 1/3 + 1/4 + 1/5 + 1/6 + ...
-->1/7 + 1/8 + 1/9 + 1/10 + 1/11 + 1/12
 q  =

    2.1032107
```

```
-->A = [1 2 3 4 5
-->6 7 8 9 10
-->11 12 13 14 15]
 A  =

    1.    2.    3.    4.    5.

    6.    7.    8.    9.    10.

    11.   12.   13.   14.   15.
```

Полиномы

- Вы столкнетесь с полиномами, например, если вы используете анализ в частотной области (в пространстве состояний) в контрольной техники. Здесь s=% s является переменной, которое определяет полином «num».
- Альтернативно, часто используется форма определения переменных s = (0, 'S')
  Полиномы могут быть определены через их корневых векторы.
- Scilab переводит корни к их соответствующим полиномам.
- Когда мы делим num полином на den полином, Scilab представляет полное выражение общего полинома.



Корни многочленов

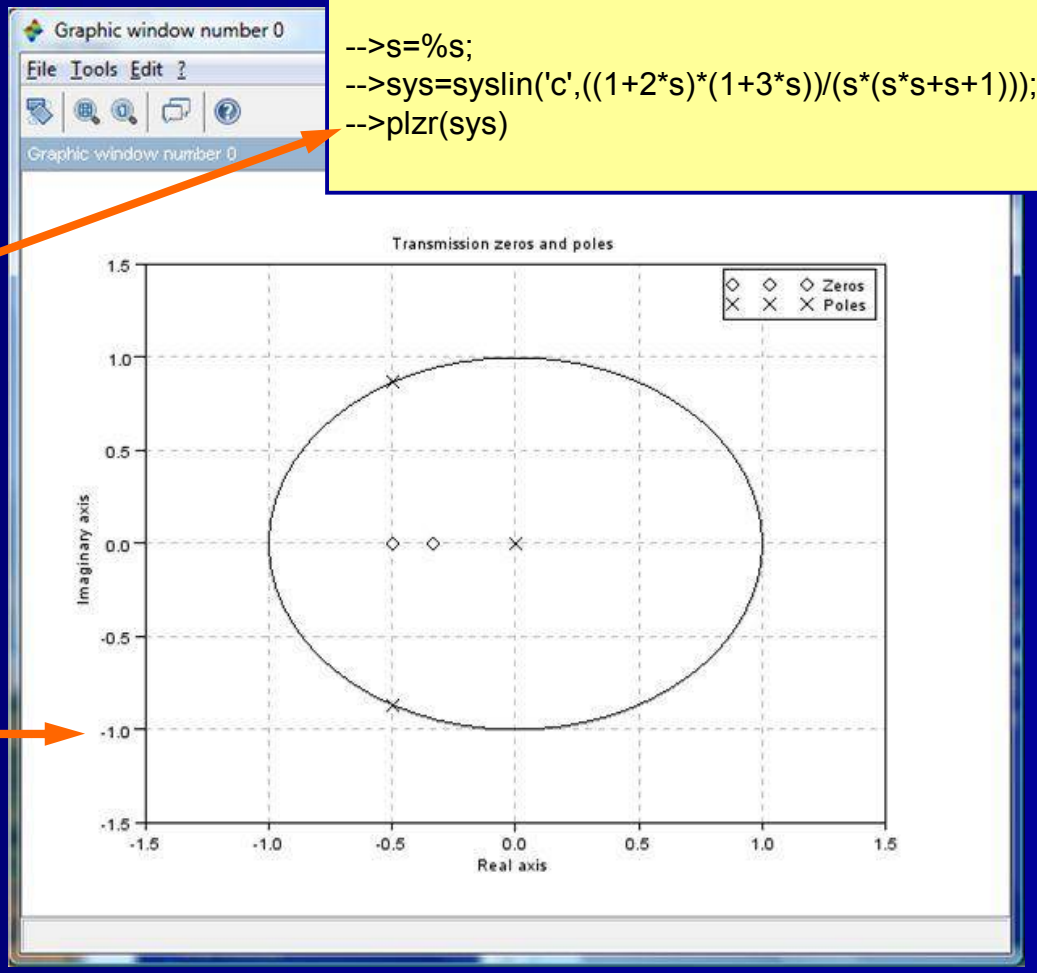
- Находить корни (нули) многочленов может быть утомительным занятием.
- Тем не менее, Scilab имеет удобный инструмент для выполнения этой задачи в виде корней (функции).
- Справа многочлены, определенные на предыдущем слайде.
- Обратите внимание, что переменные s=% s должны быть определены также.

```
-->s=%s;
-->x=roots(2*s+3*s^2+s^3)
 x  =
    0
  - 1.
  - 2.
```

```
-->s=%s;
-->z=roots(6.5625+22*s+21.5*s^2+8*s^3+s^4)
 z  =
  - 0.5
  - 1.5
  - 2.5
  - 3.5
```

Полюса и нули: plzr ()

- Функция plzr () определяет участки нулей и полюсов многочлена.
- Функция syslin () используется здесь, но будет обсуждаться позже.
- При нажатии Enter после команды (SYS) в plzr, Графика Открывающиеся окна и отображает участок.



Гаджеты (1/2): календарь

Среди встроенных гаджетов Scilab является календарь. Команда

calendar ()

возвращает календарь на текущий месяц, команду

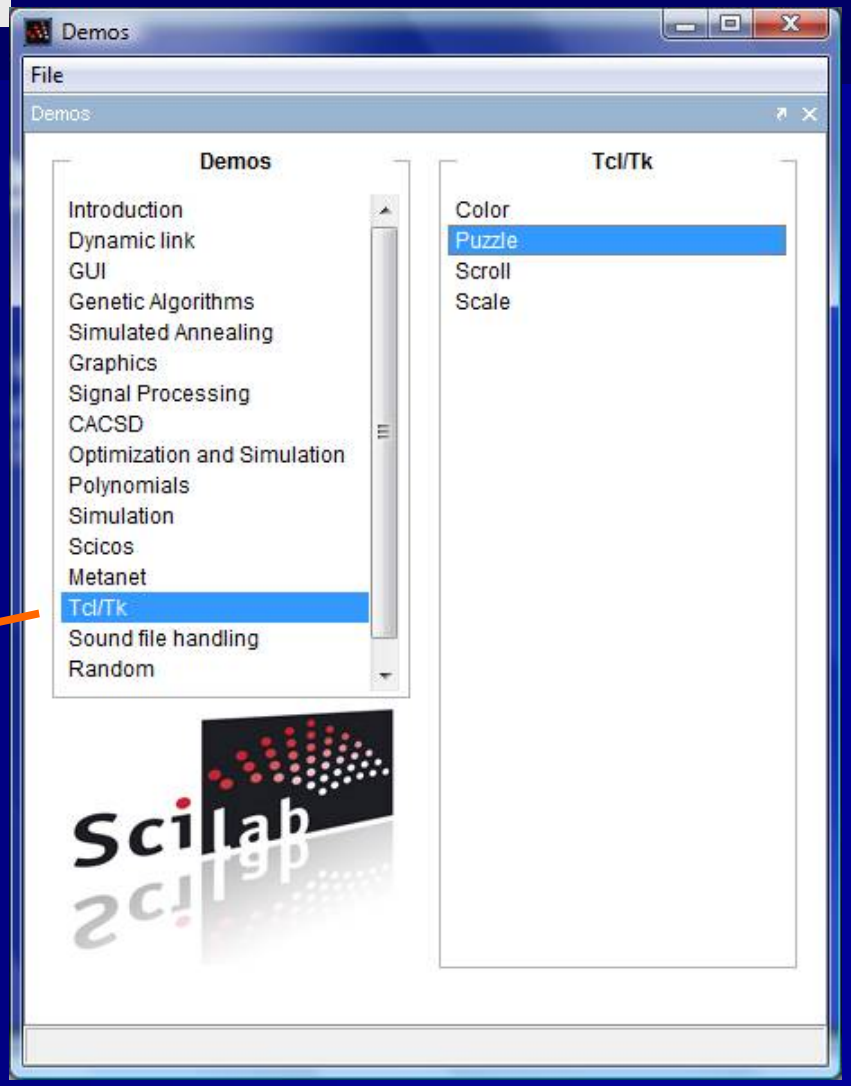
calendar (y, м)

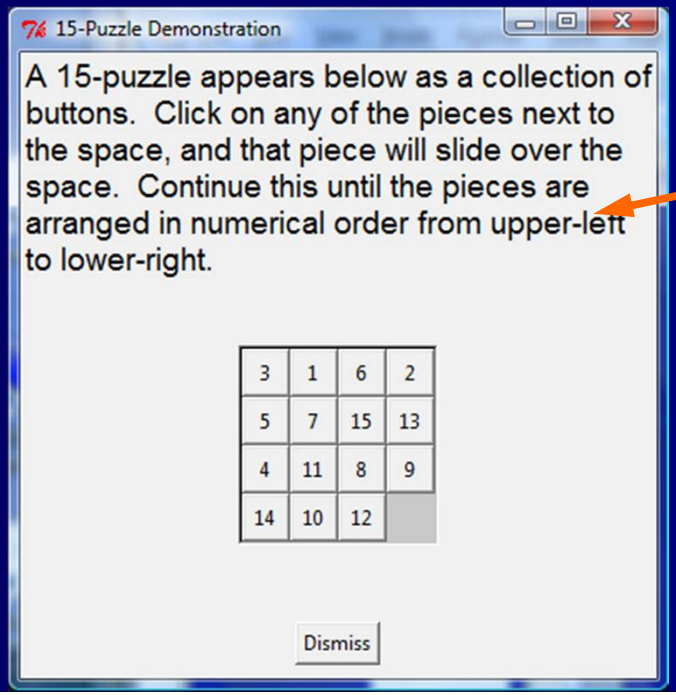
возвращает календарь на год и месяц, в случае (как показано на июнь 2013 года)

```
-->calendar(2013,6)
 ans  =

        ans(1)

Jun 2013

        ans(2)

    M     Tu    W     Th    F     Sat   Sun

        ans(3)

    0.    0.    0.    0.    0.    1.    2.
    3.    4.    5.    6.    7.    8.    9.
    10.   11.   12.   13.   14.   15.   16.
    17.   18.   19.   20.   21.   22.   23.
    24.   25.   26.   27.   28.   29.   30.
    0.    0.    0.    0.    0.    0.    0.
```

Гаджеты (2/2): головоломка

Другой гаджет представляет собой головоломку, что можно найти в демонстрациях \ Tcl / Tk \ Puzzle

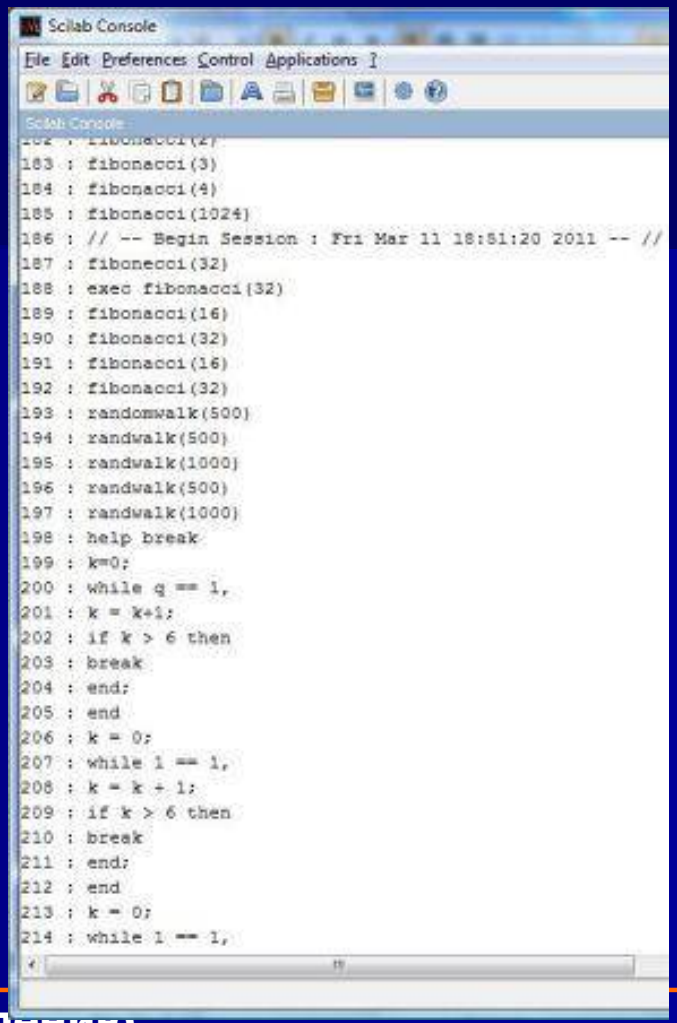




Scilab шпиона: история

Программное обеспечение, которое мы устанавливаем на наших компьютерах, как правило, шпионит за нами, собирая информацию о том, что мы делаем. Вы когда-нибудь заботились, чтобы выяснить, сколько, например, данных Index.dat Windows 'сохранил о ваших компьютерных?

Scilab шпионы с (по крайней мере) его руководителем истории. Вы можете получить доступ к этим данным, введя displayhistory () на консоли. Файл может быть очищен с помощью преференций \ Очистить историю



К сожалению,
Я не мог скопировать и вставить выписку, потому что PowerPoint неоднократно вылетал(это случается с Биллом Гейтсом, а ... Часто.)

Dr. EW
Johnny Heikell

4.
Примеры №1

Демонстрация основных программ Scilab

Пример 1-1: Сценарий для простого сюжета

- Давайте подробнее остановимся на примере из "Scilab за 15 минут"
  Мы работаем с редактором, используя тот же сценарий, как и раньше, но с добавлением комментариев
  Сохраните файл. Я называл eeo plot1.sce и сохранил его на моем USB флэш-накопитель (вы можете сохранить его где угодно)
  Чтобы запустить сценарий, нажмите на Execute значок редактора
  Что происходит показано на следующем слайде

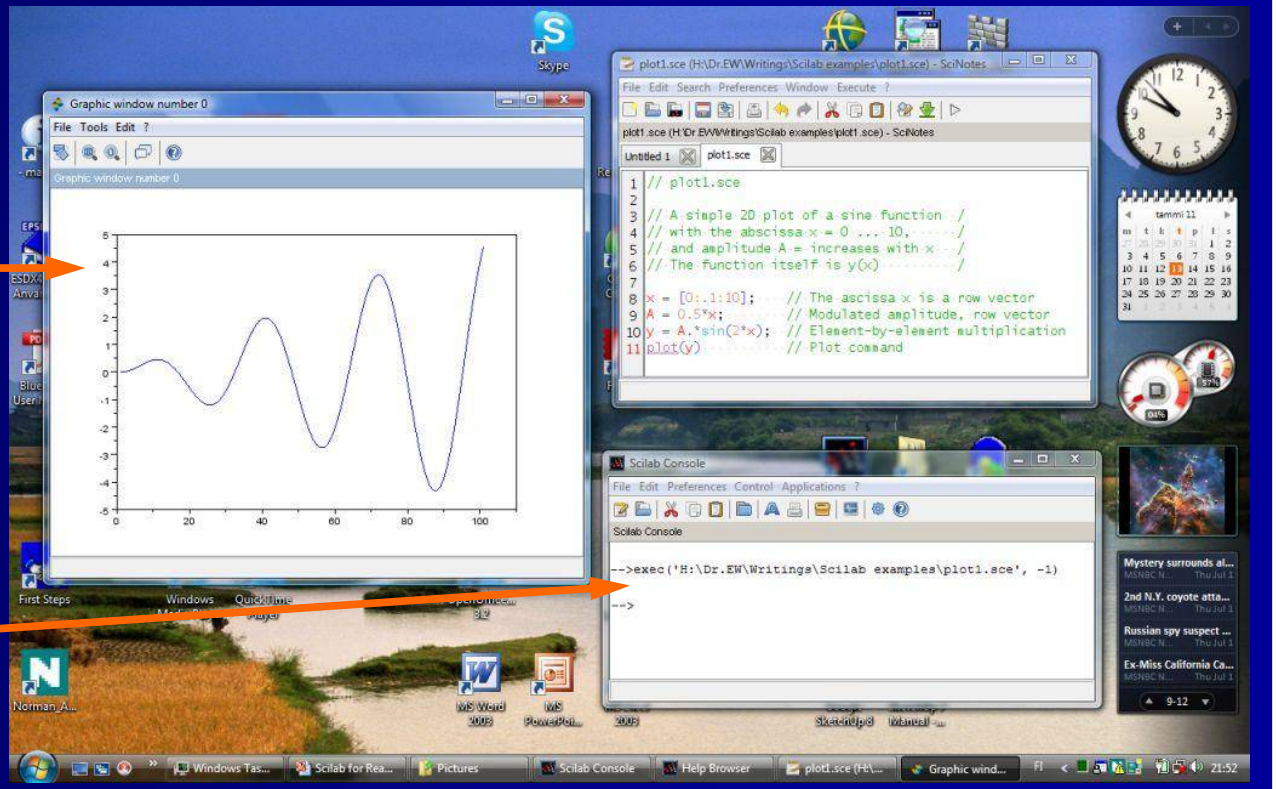
```
// plot1.sce

// A simple 2D plot of a sine function    /
// with the abscissa x = 0 ... 10,        /
// and amplitude A = increases with x     /
// The function itself is y(x)            /

x = [0:.1:10];      // The abscissa x is a row vector
A = 0.5*x;          // Modulated amplitude, row vector
y = A.*sin(2*x);    // Element-by-element multiplication
plot(y)             // Plot command
```

Примечание: Комментарии начинаются с двойной косой черты (/ /). Scilab игнорирует все позади / / когда он выполняет программу.

Экс 1-1: Графическое окно

Как вы уже поняли, Scilab использует третье окно, с графиком, чтобы представить сюжет.

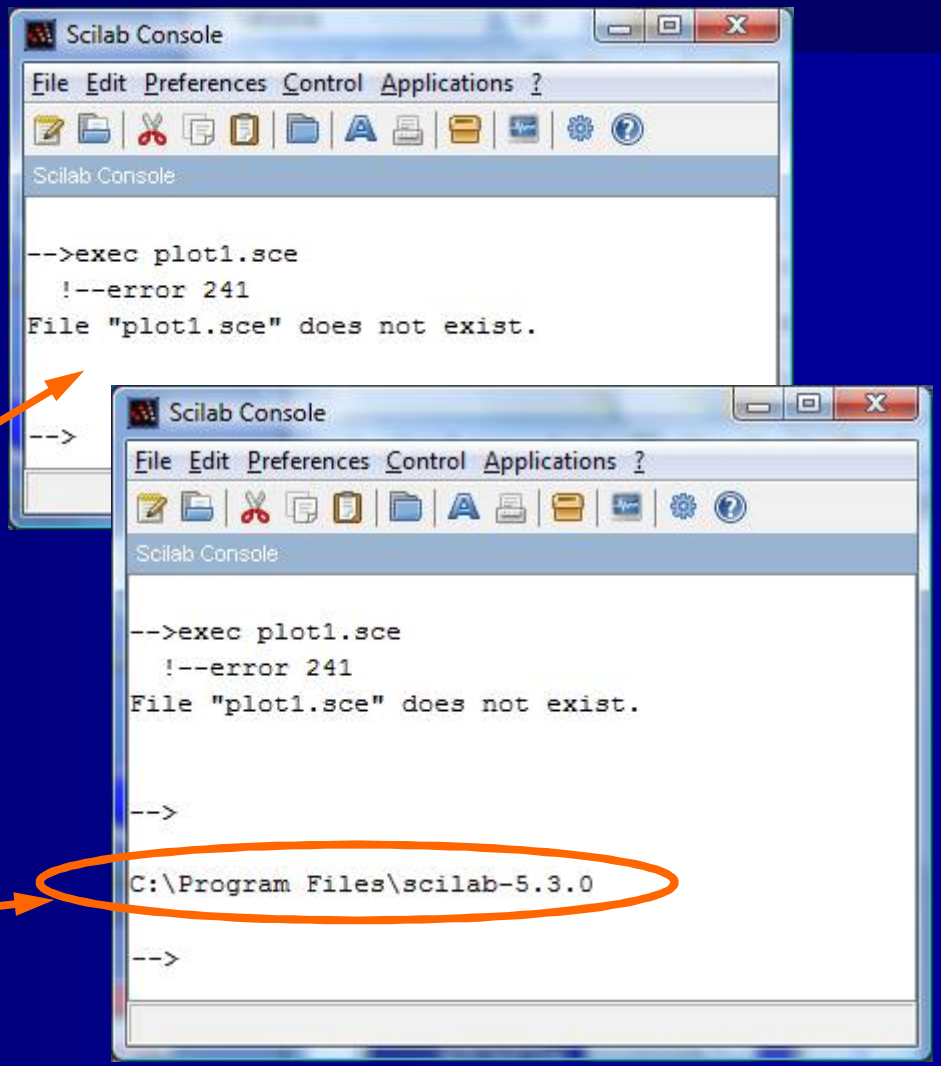
Информация о исполняемом сценарии повторяются в консоли. Сообщения об ошибках также отображаются на консоли.



Эти три окна, что мы в основном работают с, но есть и более. Вы уже видели несколько, как справки браузера и переменной браузера

Пример1-1: С помощью консоли

- Сценарий также может быть выполнен из консоли
  После командной строки введите Exec plot1.sce
  И в результате «сообщение об ошибке»
  В чем причина? Scilab ищет plot1.sce в неположенном месте
  Чтобы увидеть, где Scilab искал, Нажать: текущий каталог файла \ Display
  Ответ показан в нижнем окне:
  Это выглядит в файле программы Scilab, которая не то, где я сохранил его



Пример 1-1: изменение каталога

- Нажмите на иконку

Изменить текущий каталог ...

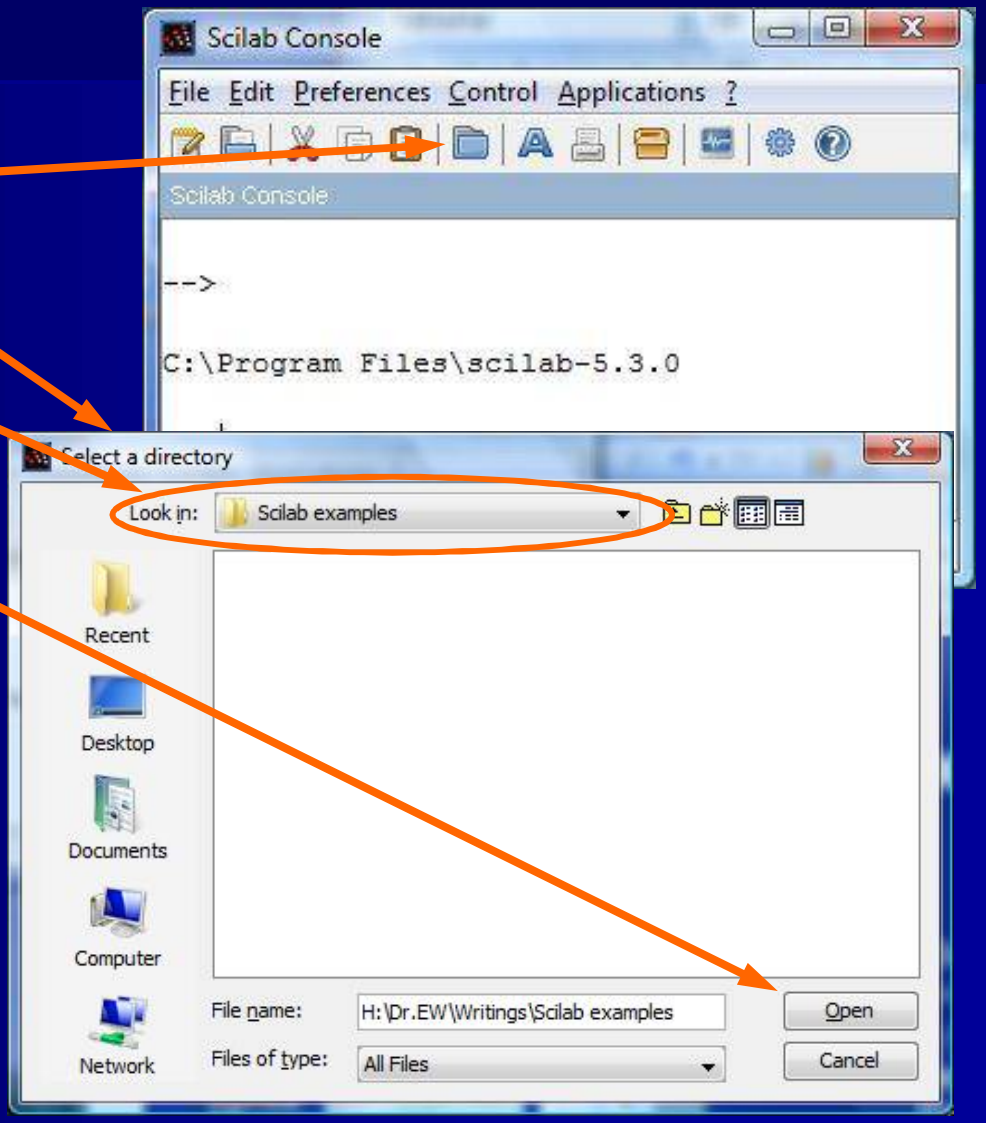
Новое окно

Определить правильный файл с выпадающем меню

Нажмите: Открыть.

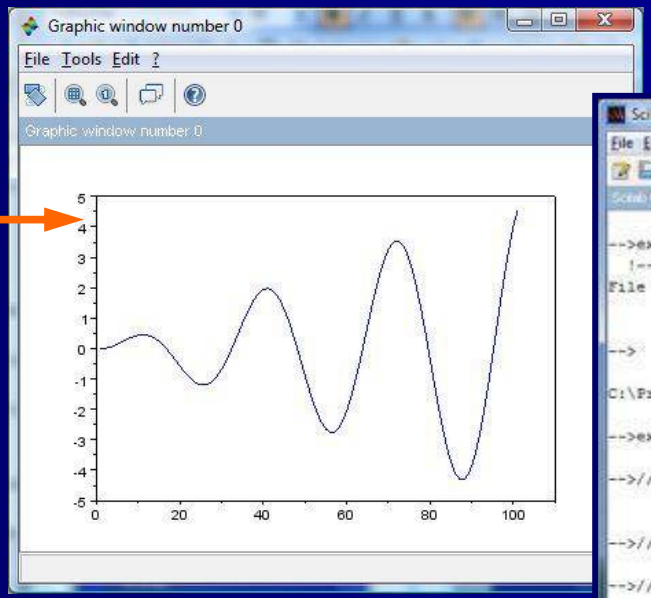
После этого можно вернуться к консоли и введите команду Exec plot1.sce

И это работает, как это видно на следующем слайде

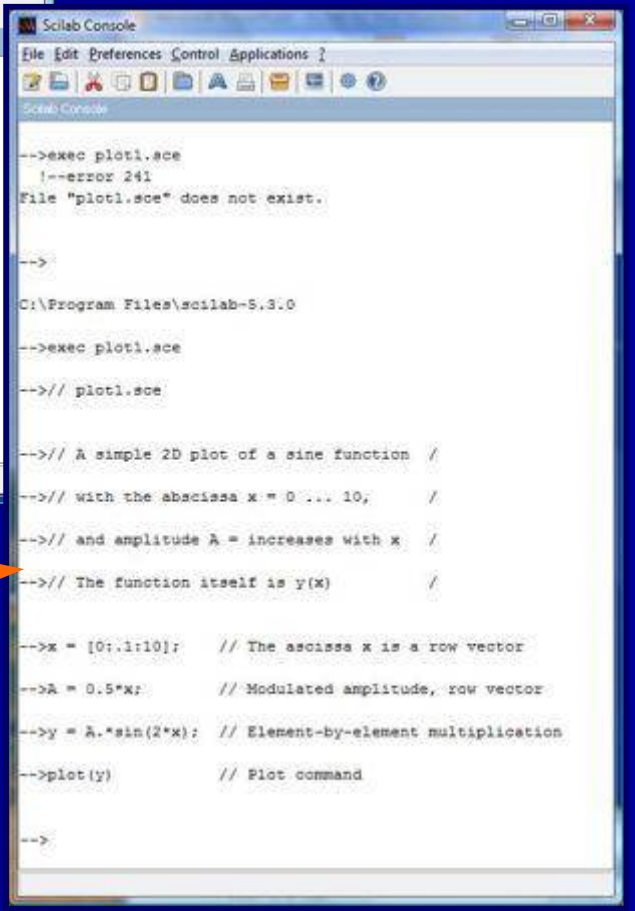


Пример 1-1: сюжет и эхо

До появления графического окна с участком определенной функции



сценарий будет выведен на командном окне (консоли)



Пример 1-1: Комментарии (1/4), детали команды

Содержание Редактор отныне будет показан на светло-зеленом фоне

- Вектор определен x = [0:0.1:10] это можно интерпретировать как "от 0 до 10 с шагом 0,1" . Умножение на Dot оператора (. *) Надо сказать Scilab следует умножить векторы элемент-на-элемента. Изменение обычным умножением (*), и вы получите сообщение об ошибке на консоли

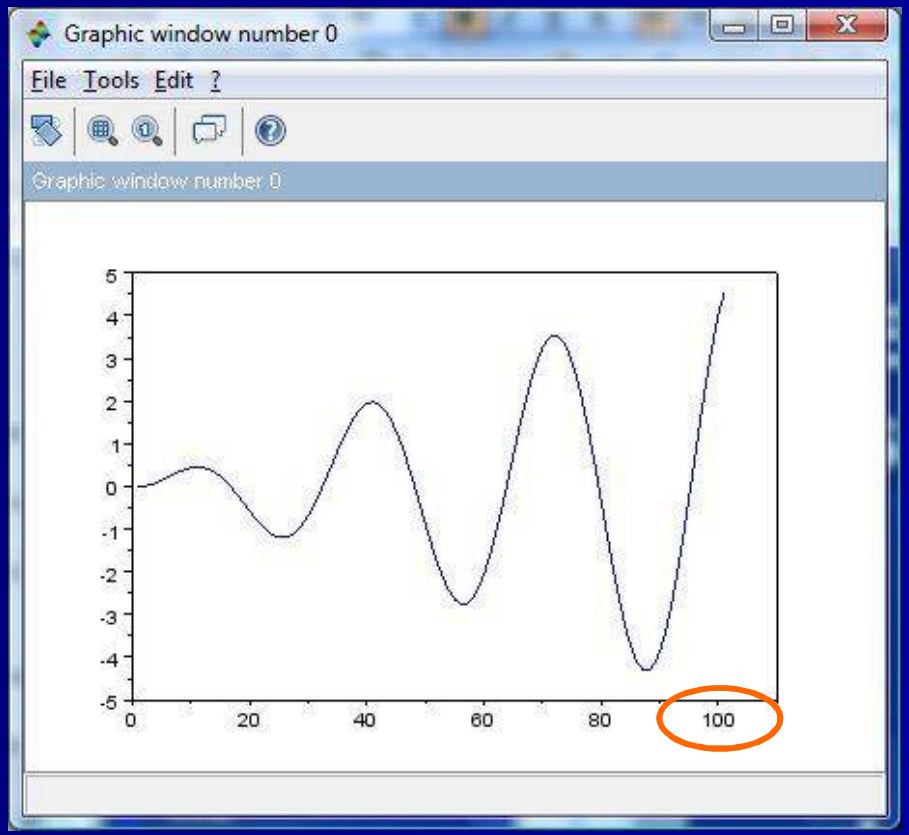
```
// plot1.sce

// A simple 2D plot of a sine function    /
// with the abscissa x = 0 ... 10,        /
// and amplitude A = increases with x     /
// The function itself is y(x)            /

x = [0:.1:10];      // The abscissa x is a row vector
A = 0.5*x;          // Modulated amplitude, row vector
y = A.*sin(2*x);    // Element-by-element multiplication
plot(y)             // Plot command
```

```
-->exec('H:\Dr.EW\Writings\Scilab examples\plot1.sce', -1)
y = A*sin(2*x);  // Element-by-element multiplication
                 !--error 10
Inconsistent multiplication.

at line      10 of exec file called by :
exec('H:\Dr.EW\Writings\Scilab examples\plot1.sce', -1)

-->
```

Пример 1-1: комментарии (2/3), сценарии

Основной сценарий, не имеет значения, метки осей, или сетку. Мы вернемся к ним в следующем примере.

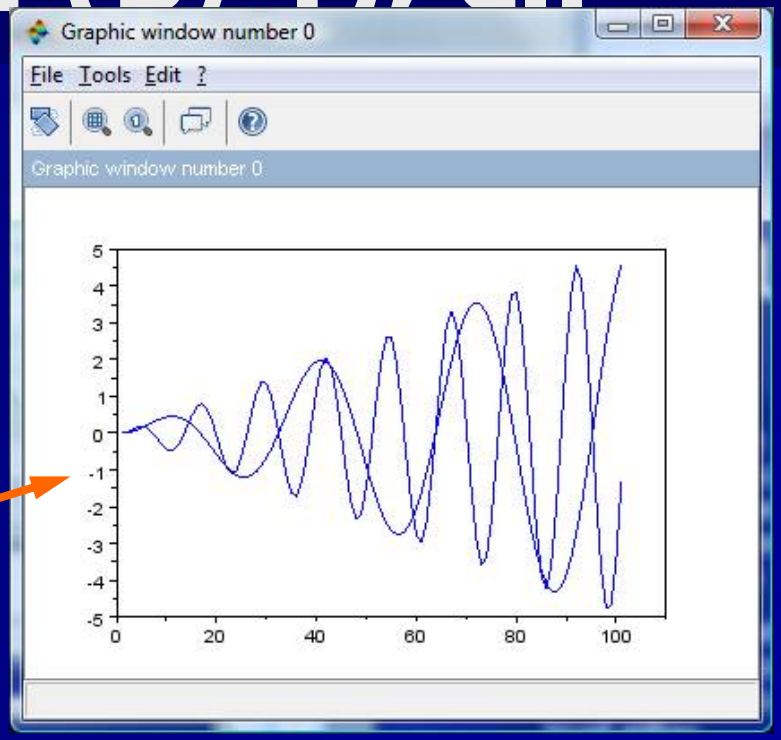
Масштаб по оси абсцисс может показаться странным, максимальное значение x было 10, но масштаб стремится к 100. На рисунке 100 на самом деле число вычислений, так как они были сделаны с шагом 0,1 до 10. Попробуйте изменить x = [0:0.2:10], и вы увидите, что шкала заканчивается на 50 (измененный сценарий должен быть сохранен, прежде чем он может быть запущен)



Пример 1-1: комментарии (3/4), clf

Предположим, что мы внесли изменения в сценарий, например, увеличить частоту (5 * S), сохраните его, и выполнить его сразу же после предыдущего запуска

В результате Scilab вычерчивает новый график поверх предыдущего. Чтобы избежать этого, мы должны либо:
- Закройте графического окна вручную после каждого запуска, или
- Добавить функцию CLF В сценарий, чтобы Scilab очистил окно.




```
clf;
x = [0:.1:10];
A = 0.5*x;
y = A.*sin(5*x);
plot(y)
```

Экс 1-1: Комментарии (4/4), уборка «мусора»

Некоторые программисты предпочитают для защиты от различных форм старых записей, которые могут повлиять на выполнения сценария. Чтобы сделать это, три команды добавляются в начале сценария:

CLEAR , удаляет элементы из рабочее пространства и освобождает память;

CLC, очищает консоль;

CLF, сбросить или восстановить текущее графическое окно

```
// plot1.sce

// A simple 2D plot of a sine function    /
// with the abscissa x = 0 ... 10,        /
// and amplitude A = increases with x     /
// The function itself is y(x)            /

clear, clc, clf;
x = [0:.1:10];      // The abscissa x is a row vector
A = 0.5*x;          // Modulated amplitude, row vector
y = A.*sin(2*x);    // Element-by-element multiplication
plot(y)             // Plot command
```

Таким образом, наша конечная сценарий выглядит следующим образом. Обратите внимание на точку с запятой (;) в конце каждого выражения, кроме последней

***) Осторожнее с CLF, это может привести к хаосу в некоторых случаях (там будет демо об этом позже)**

Пример 1-2: задача, раскладывающихся линейных функций

- Написать сценарий для линейно частотной модуляции синусоидального сигнала S (T), т.е. линейным типом

  S (T) = A (T) · sin {[2п (f0 + к (т) т] + φ}, где к скорости изменения частоты.
- Используйте 2,5 периода основной частоты
- Амплитуда должна экспоненциально затухать, $A(t) = 2e^{-t/3}$
- Начальная фаза сдвиг φ должны быть п / 4
  Построить и распечатать результат методом черчения, который отличается от предыдущего.
  Сценарий имеет сетки, название, и метки осей

Подключите в сюжете команд (), histplot (), surf (), и plot3d () на консоли, чтобы посмотреть примеры Scilab участков. См. также главу 7.

Пример 1-2: первый повтор

- Функция LINSPACE () создает линейно пространство задает вектор с аргументами из, к, числа точек. Значение по умолчанию составляет 100, но более необходимы.
  Вот Dot оператора (. *) Снова Plot2d () производит 2D участок. Аргументы t и c расшифровывается x и y топоров, число 5 выводит красный график

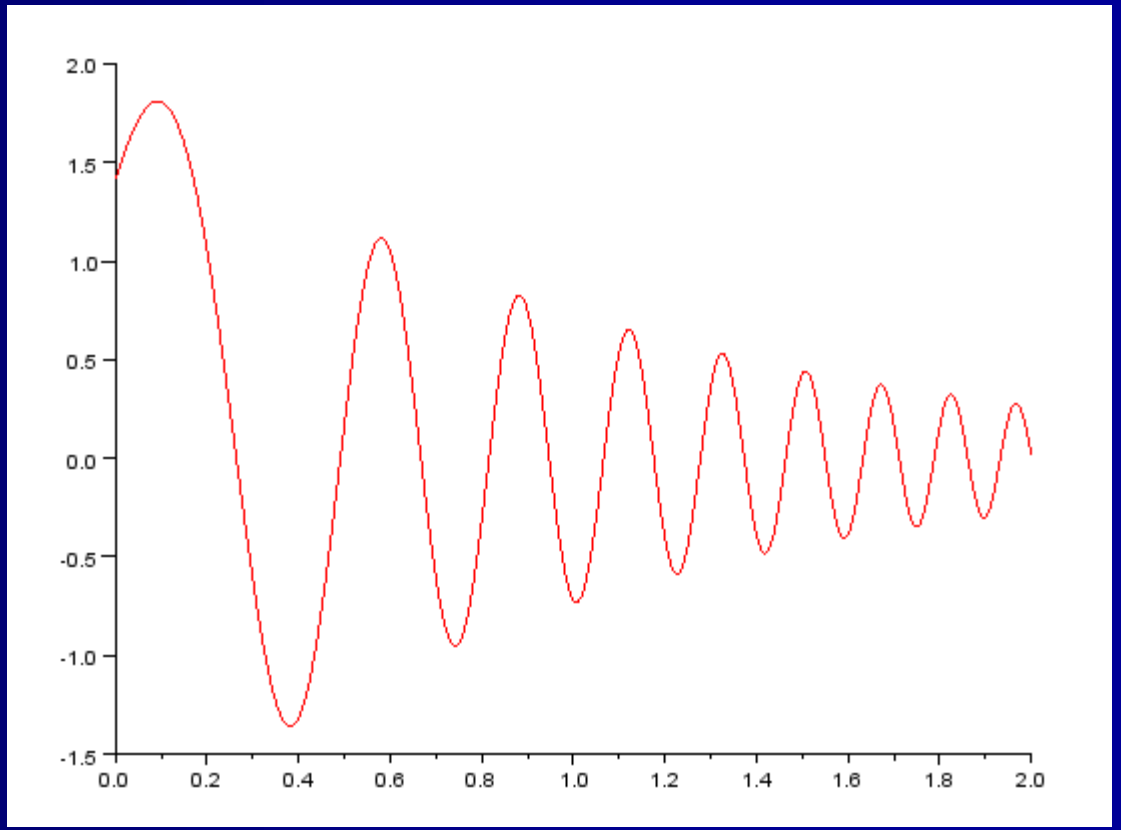
```
// f-modulation1.sce                                    /

// Plots a sinusoidal function of the type              /
// s = A(t)(sin(wt+x(t)+phi)), where w = angular        /
// velocity, x(t) = frequency modulation,  phi =        /
// phase shift, and A(t) = amplitude                    /

clear, clc, clf;
f = 1;                          // Frequency
w = 2*%pi*f;
phi = %pi/4;                    // Initial phase shift
fin = (4*%pi)/w;                // End of plot
t = linspace(0,fin,1000);
A = 2*exp(-t);
s = A.*sin(w*t + 10*t^2 + phi);
plot2d(t,s,5)
```

Примечание: Dot, используется в качестве конце имени переменной, потому что конец участка зарезервировано (Scilab ключевое слово)

Экс 1-2: график

Сюжет выглядит с начальным фазовым сдвигом, но ему не хватает сетки, названия, и меток осей

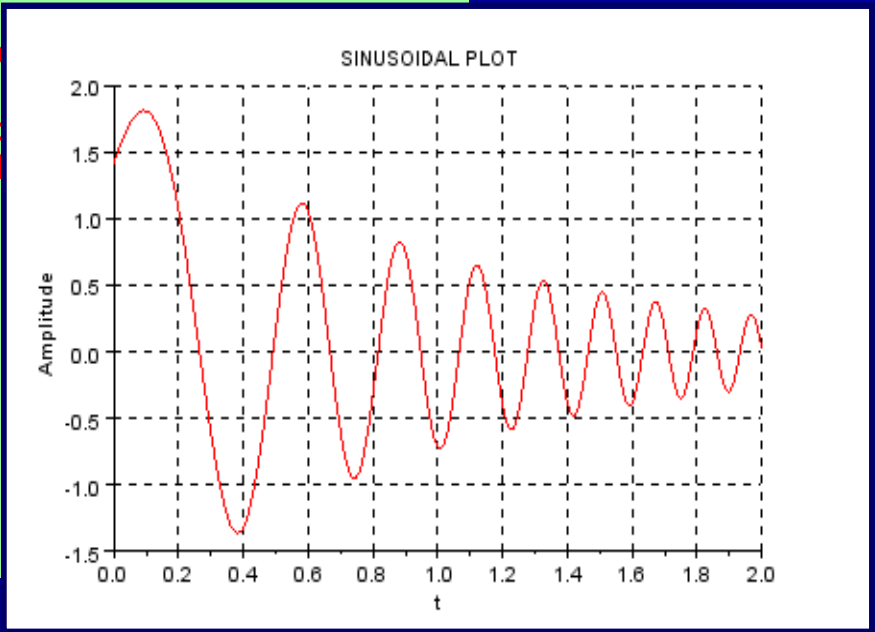
plot2d () является более универсальным, чем функция LINSPACE (), который похож на график функции в Matlab



Пример 1-2: преобразование графика

Здесь я добавил код для построения сетки, Xgrid (), название, xtitle (), а оси x и y , xlabel (), ylabel (). Сыро, но это работает

```
// f-modulation2.sce

// Plots a sinusoidal function of the type              /
// s = A(t)(sin(wt+x(t)+phi)), where w = angular  /
// velocity, x(t) = frequency modulation,  phi =     /
// phase shift, and A(t) = amplitude                    /

clear, clc, clf;
f = 1;                          // Fre
w = 2*%pi*f;
phi = %pi/4                     // Init
fin = (4*%pi)/w;                // End
t = linspace(0,fin,1000);
A = 2*exp(-t);
s = A.*sin(w*t + 10*t^2 + phi);
plot2d(t,s,5)

xgrid()
xtitle('SINUSOIDAL PLOT')
xlabel('t')
ylabel('Amplitude')
```

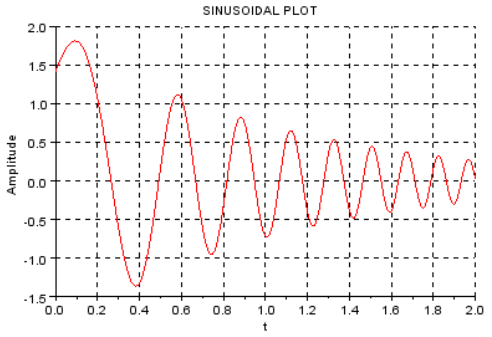


Пример 1-2: Печать

- Окна Scilab (консоль, редактор, графическое окно) имеют нормальные и расширенные функции печати.
  Один из способов получения согласованного печатного документа является: скопировать содержимое окна и вставить их в страницу текстового редактора.
  Изображение, показанное здесь было сделано на MS Word. Затем он был напечатан, как. PDF, и сохранен. Файл PNG, и, наконец, обрезаны с MS Picture Manager
  Это утомительно метод. Обратитесь за помощью передовых возможностей печати.

DECAYING LINEAR CHIRP IN SCILAB

```
// f-modulation2.sce                                    /

// Plots a sinusoidal function of the type              /
// s = A(t)(sin(wt+x(t)+phi)), where w = angular        /
// velocity, x(t) = frequency modulation,  phi =        /
// phase shift, and A(t) = amplitude                    /

clear, clc, clf;
f = 1;                          // Frequency
w = 2*%pi*f;
phi = %pi/4;                    // Initial phase shift
fin = (4*%pi)/w;                // End of plot
t = linspace(0,fin,1000);
A = 2*exp(-t);
s = A.*sin(w*t + 10*t^2 + phi);
plot2d(t,s,5)
xgrid()
xtitle('SINUSOIDAL PLOT')
xlabel('t')
ylabel('Amplitude')
```



Пример 1-2: проверка

Чтобы показать, что частота линейно модулированной, можно добавить частоту в зависимости от t до участка
Для этого мы добавляем функцию f_mom.
Команда участок также должен быть изменен. мы
перейти к командной () участка и включить оба параметра вместе с информацией о цвете ('R', 'б')
Предохранитель x-лейбл ' t ' в качестве аргумента xtitle ()
обменять y-метку для легенды ();
аргумент 2 относится к верхнему левому углу

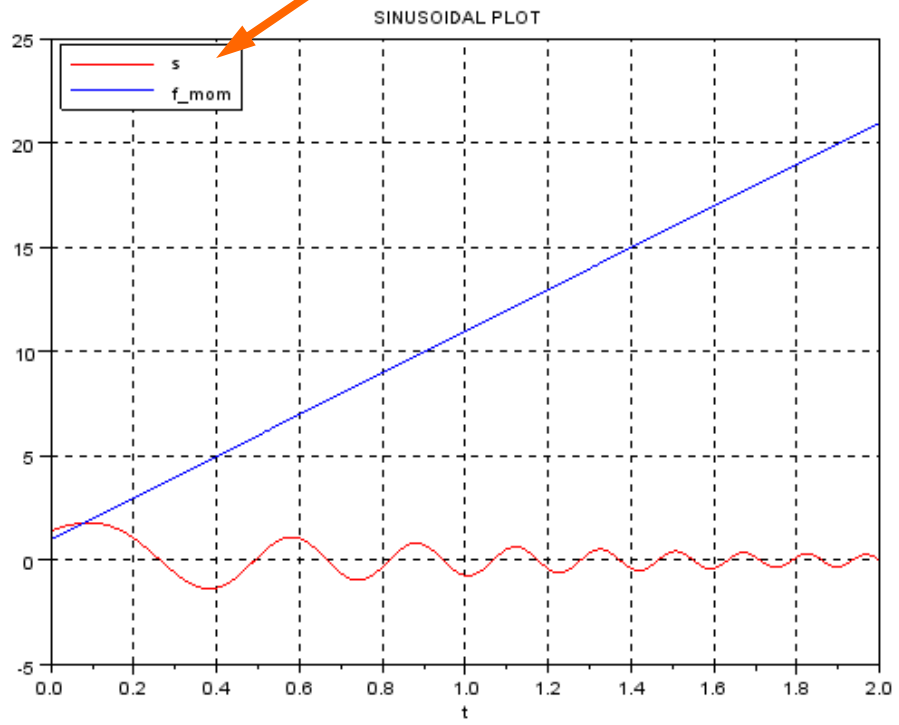
```
// f-modulation3.sce                                    /

// Plots a sinusoidal function of the type              /
// s = A(t)(sin(wt+x(t)+phi)), where w = angular        /
// velocity, x(t) = frequency modulation, phi =         /
// phase shift, and A(t) = amplitude. Second            /
// plot for momentary frequency values                  /

clear, clc, clf;
f = 1;                          // Frequency
w = 2*%pi*f;
phi = %pi/4;                    // Initial phase shift
fin = (4*%pi)/w;                // End of plot
t = linspace(0,fin,1000);
A = 2*exp(-t);
s = A.*sin(w*t + 10*t^2 + phi);
f_mom = f + 10*t;               // Momentary frequency
plot(t,s,'r',t,f_mom,'b')
xgrid()
xtitle('SINUSOIDAL PLOT','t')
legend('s','f_mom',2)
```

Экс 1-2: Конечный график

Он не является оптимальным участком, но информация есть.

С большими различиями в вертикальных шкалах, мы должны либо использовать логарифмическую ось y или отделить одну сюжетную линию от другой, что приходит позже

Обратите внимание на легенде



Пример 1-2: обсуждение

- Как было сказано ранее, Scilab развивается со временем и похож на Matlab с каждым выпуском
  В качестве примера в случае Scilab в Помощь Браузер признает xlabel () и ylabel (), который я использовал в улучшение графика, как функции Matlab, а также называет их функций Scilab
  Тем не менее, есть много устаревших функций Scilab, и вы найдете их на всем протяжении, если будете полагаться на старые учебники. Даже Scilab в Помощь Браузер может ссылаться на них.
  Будьте осторожны, особенно если имя функции начинается с x (см. примечание в главе 7)
  Возможно, вы заметили, что я начинаю сценарий с комментарием с указанием имя скрипта (например, / / ж-modulation3.sce /). Я делаю это, чтобы помочь определить сценарий, когда я смотрю распечатки.

Пример 1-3: Лото, задача

Первая часть этого примера заимствовано из учебника Mäkelä

Задача 1: Создать определенную функцию пользователя (UDF), которая рисует ряд чисел лото. Предположим, что Лото строка содержит 7 номеров, 1-39

Задача 2: Написать сценарий, который вызывает предыдущую функцию (или модификацию этого, если необходимо) и производит участок для его визуальной индикации, если функция производит случайные числа. Генерация 10000 рисует для выполнения этой задачи

Пример 1-3: задача 1, сценарий

- dt=getdate()
- возвращает ДД-ММ-ГГГГ
  rand('seed',n) устанавливает случайный ген-ратор чисел n
  dt(9) ставит число между 00 и 59, dt(10 возвращает миллисекунды 000 ... 999
  В то время как ... В конце конструкции будет подпадать под обсуждении ниже

Функция ID, не комментарий

```
функция lotto

//---------------------------------------------------------------/
/ / Функция ничьих 7 номера лото [1,39], сначала /
/ / Создание семя используя текущую дату и время /
/ / (Второй, миллисекунды) информация /
//---------------------------------------------------------------/

dt=getdate(); / / Pick текущую дату
 rand('seed',1000*dt(9)+dt(10)); / / Инициализация генератора
numbers=floor(1+39*rand(1,7)); / / Рисуем   Лото строку
 while(length(unique(numbers))<7)
numbers=floor(1+39*rand(1,7)); / / Если число повторяется в строке,
    end
                                / / то привлекается новая     строка
 numbers=gsort(numbers); / / Сортировка чисел в порядке убывания
 disp(numbers(7:-1:1)); / / Показать в порядке возрастания
endfunction
```

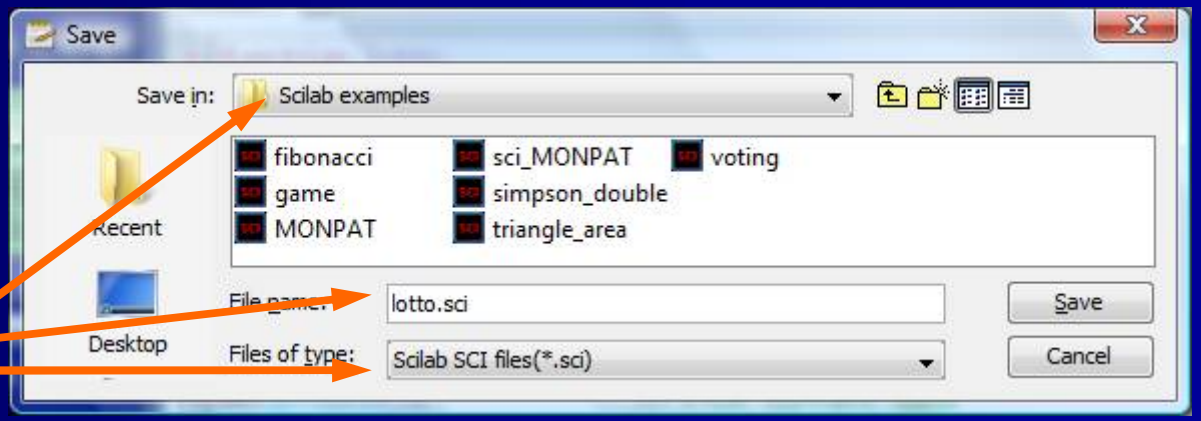
Пример 1-3: задача 1, экономия

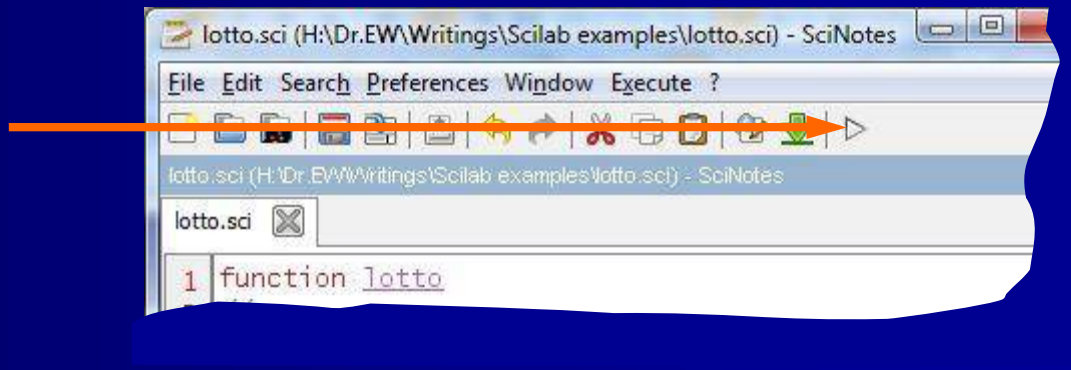
Этот сценарий (функция) несколько отличается от предыдущих, так что давайте идти через операции сохранения:

Сохраните сценарий как lotto.sci в нужную папку

Затем кликните по значку Выполнить и загрузить сохраненный файл в Scilab

...





Пример 1-3: задача 1, бег

Если консоль показывает предупреждение, свяжитесь с Помощью, то это может значить следующее: Это может быть проигнорировано или funcprot (0) команда может быть добавлен в сценарий, чтобы избежать предупреждение. Вы также можете перейти к главе 18 для краткого объяснения

Выполните (ход) загруженной функции, введя имя функции на консоли

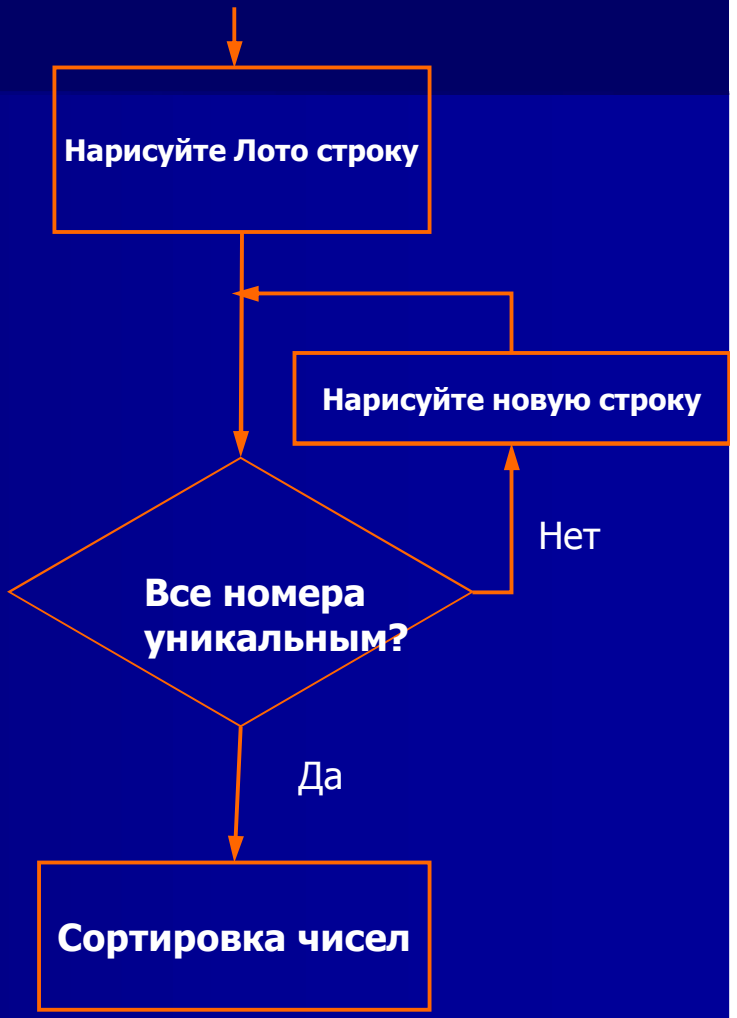
```
-->exec('H:\Dr.EW\Writings\Scilab examples\lotto.sci', -1)

Предупреждение: переосмысление функции: лото
. Используйте funcprot (0), чтобы избежать этого сообщения
```

```
-->exec('H:\Dr.EW\Writings\Scilab examples\lotto.sci', -1)

Предупреждение: переосмысление функции: лото
. Используйте funcprot (0), чтобы избежать этого сообщения

-> помощь funcprot

-->lotto
  3.   5.   13.   15.   33.   37.   39.
```

И выигрышные номера являются ...

Пример 1-3: задача 1, обсуждение

- Это упражнение программирования уже промежуточный уровень. Не волнуйтесь, если это вызывает у вас проблемы. Большинство его деталей будет повторено позже
- Обратите внимание на элегантное решение для тестирования уникальности номера:
- $\text{length}(\text{unique}(\text{numbers})) < 7$
- Тем не менее, в теории оно может стать практически бесконечный цикл ...



Пример 1-3: задача 2, сценарий (1/2)

В предыдущем UDF должна быть изменена, отдельным кодом: 1) Удалить сортировки и отображения и 2) пересмотреть функции ID, чтобы выполнить

В последнем случае он имеет один или несколько входных аргументов (в), которые даются ему вызывающих командования и выходных аргументов [Out], с помощью которого он возвращает результат своих расчетов с вызывающим команды (см. следующий слайд)

```
// lotto2.sce

//-------------------------------------------------------------------/
// Сценарий просит числа Лото обращает, что мы /
/ / Хотите сделать, используя диалоговое окно отдельный. Затем
он вызывает /
/ / Локальная UDF lottodraw ()), который генерирует ряд N /
/ / Случайные числа Lotto в диапазоне [1,39]. Это сортирует /
/ / Числа в векторе, добавив один (1) соответствующая /
/ / Элемент вектора для каждого соответствующего хит.
Результат /
/ / Строится после вошел ничьи.
/
//-------------------------------------------------------------------/

clear,clc,clf;

// Функция (подпрограммы) lottodraw ():
/ / ---------------------------------------------- -----
   / / Функция рисует N Лото номера [1,39], с /
   / / N определяется через входной аргумент дюйма /
   / / Он обеспечивает нарисованный строку в вызывающий
сценарий /
   / / Команда через выходной аргумент вне./
   / / Случайность нарисованных цифр улучшается /
   / / Сначала создать семя, используя текущую дату и /
   / / Время (второй, миллисекунды) информация. /
```

Пример 1-3: задача 2, сценарий (2/2)

Redefined функции (подпрограммы)

Количество Лото вводится через отдельный x_dialog в диалоговом окне ()

Вытянутые номера Лото собраны в векторе столбцов внутри цикла

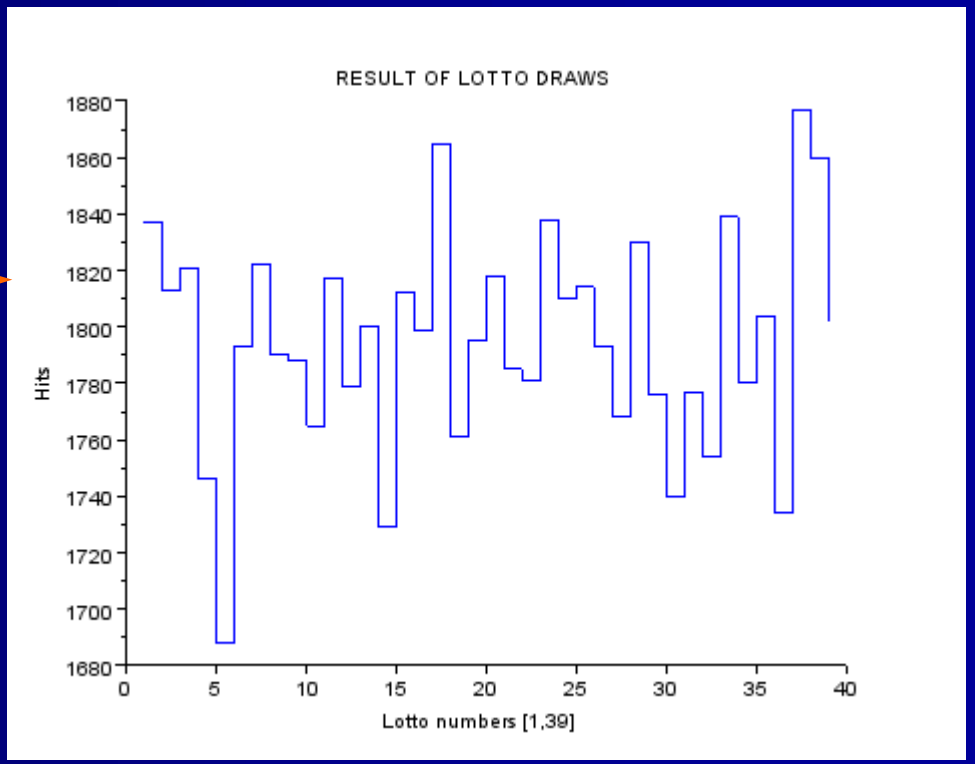
В результате на графике как ступенчатых функций

```
function out=lottodraw(in)
   dt=getdate(); // Pick current date
   rand('seed',1000*dt(9)+dt(10)); // генератор случайных  чисел
   out = floor(1+39*rand(1,in)); Рисуем Лото строку (из переменной) )
   while(length(unique(out))<in)    // Если число повторяется в строке, ,
      out = floor(1+39*rand(1,in)); то новая строка обращается
   end
endfunction

// (MAIN) Call subroutine, update histogram, plot:
//---------------------------------------------------------------
M = evstr(x_dialog('Enter # of... Открыть диалоговое окно
lotto draws ',''));
N = 7;                                  // Лото нарисовать
columns = zeros(1,39);                  // Инициировать сбор вектор
for k = 1:M
   numbers = lottodraw(N);              // Призыв к подпрограмме
   columns(numbers)=columns(numbers)+1;
                                        // Добавить 1 для нарисованной числа
end

x = linspace(1,39,39);                  // Определите ось x
plot2d2(x,columns,style=2)              //
график как ступенчатых функций

xtitle('RESULT OF LOTTO DRAWS')  // Добавить название & этикетки
xlabel('Lotto numbers [1,39]')
ylabel('Hits')
```

Пример 1-3: задача 2, выполнение и график

Диалоговое окно всплывает при выполнении сценария. Введите желаемое число Лото ничьих и нажмите кнопку ОК

Результат нанесены в графическом окне. Это не так уж плохо, учитывая, что в среднем 10 000 рисует 7x10, 000/39 = 1795



Пример 1-3: комментарии (1/3)

- Это было не совсем техническая проблема, но она показал многие черты Scilab
  UDF в задаче 1 необычен тем, что были закрыты, не имея вход или выход аргументы -вы просто использовать его как есть.
  Местный UDF продемонстрировали в задаче 2 как обычно и бывает.
  В дополнение `rand()`, 1 Задача приносит несколько полезных функций: GETDATE (), `floor()`, `unique()` и gsort ()
  Сценарий в задаче 2 комментирует длину. Добавление заголовков и комментарии занимает время, и они требуют места, но комментарии являются абсолютно необходимыми, чтобы разобраться в программе на более позднем сроке.
  Задача 2 вводит диалоговое окно, в GUI (графический интерфейс пользователя) функцию, к которой мы еще вернемся в главе 15

Пример 1-3: комментарии (2/3)

- В дополнение к командам `plot()` и plot2d (), которые мы использовали, Scilab имеет множество других способов создания графиков, наряду с вариантами добавив уточняющие текстовые строки к сюжетам. Построение будут более подробно раскрыты в главе 7.
  Управление потоком, в данном случае этот термин относится к использованию условного отраслевых структур-будет обсуждаться в главе 11
  Примеры 1-1 ... 1-3 были также призваны подчеркнуть тот факт, что мы вынуждены "думать матрицу-мудрый" при работе с Scilab. Например, Scilab сразу выдает сообщение об ошибке, если мы попытаемся сделать обычное произведение (*), если параметр используется в матричной форме и требует Dot умножение (. *)
  (Напомним Пример 1-1)

Пример 1-3: Комментарии (3/3), округление функции

Округления Функция floor() является одним из четырех функций округления в Scilab: round(), fix() (or int()), floorиceil()
Обратите внимание на разницу между первым и двумя последними

round()	Округляет в сторону ближайшего целого
fix() or int()	возвращает целую часть числа
floor()	Округляет к меньшему
ceil()	Округляет к большему

```
-->round(-2.7), round(2.7)
 ans  =

  - 3.
 ans  =

    3.

-->fix(-2.7), fix(2.7)
 ans  =

  - 2.
 ans  =

    2.
```

```
-->floor(-2.7), floor(2.7)
 ans  =

  - 3.
 ans  =

    2.

-->ceil(-2.7), ceil(2.7)
 ans  =

  - 2.
 ans  =

    3.
```

Dr. EW
Johnny Heikell

5. Матрицы, функции и операторы

Обзор основных матричных операций, функций и операторов

Введение

- Так как Scilab построен вокруг матриц, мы вынуждены их использовать.
  Scilab хранит числа (и символы) в матрицах.
  Матрицу можно увидеть в виде таблицы, состоящей из t строк и p столбцов .
  Скалярные переменные не существуют сами по себе, они рассматриваются как 1x1 матриц
  Общий вид матрицы Scilab (здесь матрица 3x3) является?= [11 12 13; 21 22 23; 31 32 33] элементы строк можно также через запятую:= [11, 12, 13, 21, 22, 23, 31, 32, 33]
  На следующей странице приведены оба варианта для матрицы 3x3

"[Вектор] никогда не было ни малейшей пользы любого существа."
Приписывается лорд Кельвин

Матрица 3x3

Оба варианта для выражения матриц интерпретируются так же, по Scilab. Выберите какой вы хотели

```
-->A = [11 12 13; 21 22 23; 31 32 33]

 A  =

    11.    12.    13.
```

```
-->A = [11, 12, 13; 21, 22, 23; 31, 32, 33]

 A  =

    11.    12.    13.
    21.    22.    23.
    31.    32.    33.
```

Примечание: Scilab может привести к скопированный текст экрана (как показано здесь), чтобы подчеркнуть, при вставке в другой документ. Если это так, поставьте курсор в конце текста и нажмите Backspace ()

Строки и столбцы векторов

Задача 1: Создать вектор-строку с первого элемента 0, последний элемент 1 с шагом 0.2. Обратите внимание на порядок и двоеточия, чтобы разделить элементы

```
-->row=[0:0.2:1]

 row  =
```

Задача 2: Создайте похожий вектор-столбец. Обратите внимание на звездочку, что означает транспортирование матрицы

```
   0.    0.2    0.4    0.6    0.8    1.
-->column=[0:0.2:1]'
 column  =

    0.
    0.2
    0.4
    0.6
    0.8
    1.
```

В случае, если окно консоли установлено слишком маленьким, и все элементы не помещаются, то Scilab прерывает вывод и спрашивает, нужно ли их продолжить.

Некоторые специальные матрицы

3x3 **идентичная матрица**

```
-->C=eye(3,3)

 C  =
   1.    0.    0.
   0.    1.    0.
   0.    0.    1.
```

3x2 матрица единиц

```
-->D=ones(3,2)
 D  =
   1.    1.
   1.    1.
   1.    1.
```

2x3 нулевая матрица

```
-->E=zeros(2,3)
 E  =

   0.    0.    0.

   0.    0.    0.
```

Функция rand (t, p) создает равномерно распределенное MXN матрицу. Добавление аргумент «нормальным» создает нормальную распределенную матрицу

```
-->rand(4,4)
 ans  =

    0.2312237   0.3076091   0.3616361   0.3321719
    0.2164633   0.9329616   0.2922267   0.5935095
    0.8833888   0.2146008   0.5664249   0.5015342
    0.6525135   0.312642    0.4826472   0.4368588

-->rand(4,4,'normal')
 ans  =

  - 1.3772844  - 0.6019869  - 0.3888655  - 0.7004486
    0.7915156  - 0.0239455  - 0.6594738    0.3353388
  - 0.1728369  - 1.5619521    0.6543045  - 0.8262233
    0.7629083  - 0.5637165  - 0.6773066    0.4694334
```

Основные расчеты матричные

```
-->A = [1 2 3; 4 5 6]; B = A; C = A + B
 C  =
    2.    4.    6.
    8.    10.   12.
```

дополнение

```
-->A = [1 2 3; 4 5 6]; B = A'; C = A * B

 C  =
    14.   32.
    32.   77.
```

Умножение (примечание транспонирования!)

```
-->A=[1 2 3; 4 5 6]; B=[A]; C=A/B
 C  =

    1.          1.518D-16
    3.795D-15   1.
```

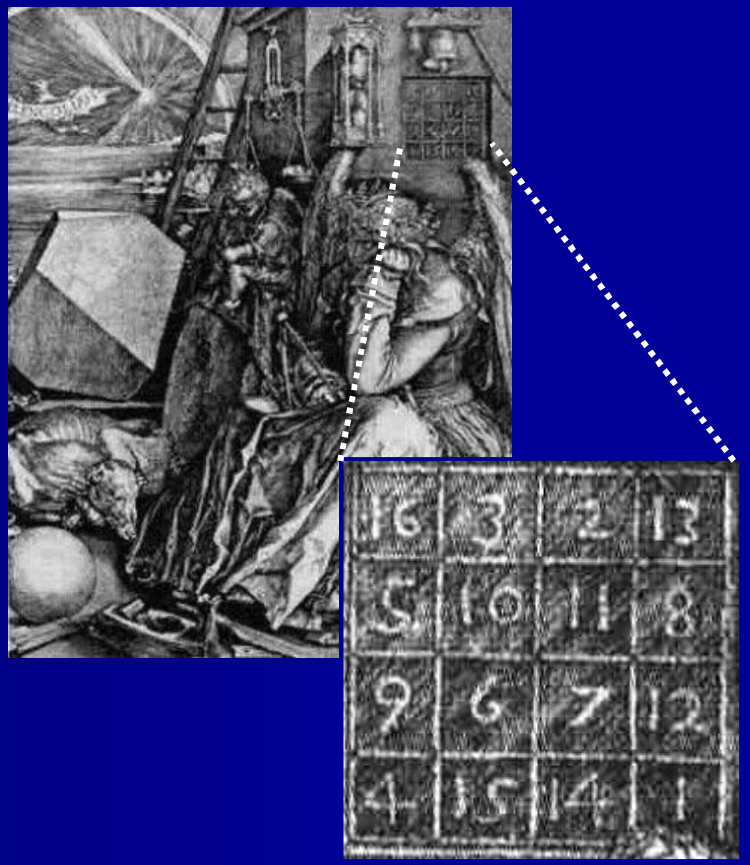
```
-->A = [2 3; 4 5]; H = inv(A)
 H  =
  - 2.5    1.5
    2.   - 1.
```

Обратная матрица

Отдел (примечание ошибок округления)
Примечание 1: Правила для матричных операций, конечно, должны быть соблюдены!
Примечание 2: Scilab возвращает D, а не электронную, для показателя (1.518D-16); точное значение равно 0, но здесь мы имеем случай ограниченной точностью вычислений

Магический квадрат Дюрера

- Немецкий художник эпохи Возрождения и любитель математики Альбрехта Дюрера придумал "волшебный" квадрат, который является популярным примером в линейной алгебре.
В квадрате Дюрера суммы выгравированных чисел любой строки, столбца или диагональную дают одинаковый результат (34)
Мы будем использовать магический квадрат, чтобы исследовать некоторые аспекты матричных операций
Магический квадрат будет обозначаться "M", чтобы установить его отдельно от других матриц .
Обратите внимание, что многие операции с матрицами определены только для квадратных матриц.



sum(), transpose, and diag()

- Магический квадрат вводится в командной строке Консоли.
- В sum(M) производит сумму всех элементов. Это отличается от Matlab, где же утверждение возвращает сумму четырех столбцов, т. е. сумма (M) = 34. 34. 34. 34.
- transpose(M) переворачивает матрицу относительно главной диагонали.
- diag(M), наконец, возвращает главной диагонали как вектор-столбец

```
-->M = [16 3 2 13; 5 10 11 8
-->9 6 7 12; 4 15 14 1]
 M  =

    16.    3.    2.    13.
    5.     10.   11.   8.
    9.     6.    7.    12.
    4.     15.   14.   1.

-->sum(M)
 ans  =

    136.
```

```
-->M'
 ans  =

    16.    5.    9.    4.
    3.     10.   6.    15.
    2.     11.   7.    14.
    13.    8.    12.   1.

-->diag(M)
 ans  =
    16.
    10.
     7.
     1.
```

Сумма строк и столбцов: sum()

- Scilab возвращает суммы строк и столбцов матрицы A с командами суммы (A, 'c') и суммы (A, 'R') соответственно
- На первый взгляд, использование 'c' и 'R' аргументов чувствует странным. Смсыл в том, что 'R' возвращает суммы столбцов матрицы, дающих вектор-строку, в то время как 'c' возвращает суммы строк матрицы, вектор-столбец
  Альтернативные утверждения: sum (`R ') = sum (A, 1) и sum (A,' C ') = sum (A, 2)

```
-->A = [1 2 3; 4 5 6; 7 8 9]

 A  =
    1.    2.    3.
    4.    5.    6.
    7.    8.    9.
-->B = sum(A,'r')

 B  =
    12.    15.    18.
-->C = sum(A,'c')
 C  =
     6.

    15.

    24.
```

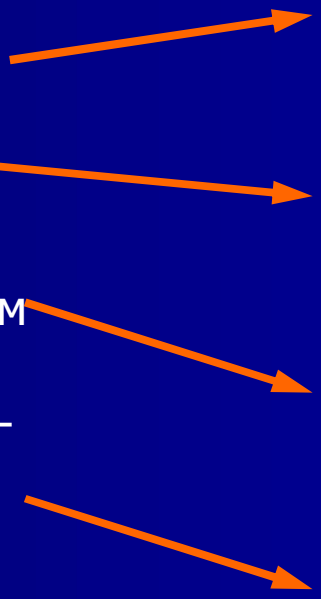
prod()

- Продукт из строк и столбцов может быть сформирован таким же образом, как и sum
- prod (, 'R') возвращает произведение каждого столбца в качестве вектора-строки
- prod (, 'C') возвращает произведение каждой строки как вектор-столбец
- Prod () возвращает произведение всех матричных элементов

```
-->A=[1 2 3; 4 5 6; 7 8 9]
A  =

    1.    2.    3.
    4.    5.    6.
    7.    8.    9.

-->prod(A, 'r')
ans  =

   28.    80.    162.

-->prod(A, 'c')
ans  =

    6.
    120.
    504.

-->prod(A)
ans  =

   362880.
```

min(), max()

- Такая же логика продолжается с функциями min () и max ().
- min () выделяет наименьший элемент в матрице, а max(A) наибольший.
- min(, 'R') возвращает вектор-строку, состоящую из мельчайших элементов в каждом столбце.
- max (A, 'c') возвращает вектор-столбец, содержащий самые большие элементы в каждой строке



```
-->A=[3 0 1; 2 2 7; 5 9 4]
A  =

    3.    0.    1.
    2.    2.    7.
    5.    9.    4.

-->min(A)
ans  =

    0.

-->max(A)
ans  =

    9.

-->min(A, 'r')
ans  =
    2.    0.    1.

-->max(A, 'c')
ans  =

    3.
    7.
    9.
```

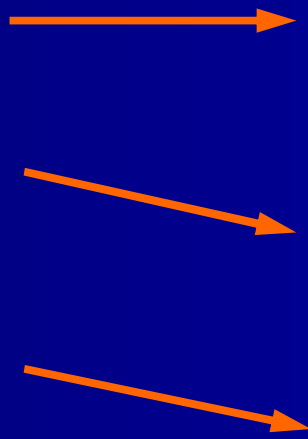
Min/max положение и значение

- Разновидность min () max() позволяет определить положение и величину наименьшего элемента.
- Самый большой матричный элемент [min_value min_pos] = min () выделяет положение и значение (в этом порядке!) наименьшего элемента в матрице, [max_val max_pos] = max (A) – наибольший.
- Примечание: Обозначение векторных элементов (здесь min_val и т.д.) не имеет значения.

```
-->A = [5 3 1; 2 4 6];
-->[min_val min_pos] = min(A)

 min_pos  =
    1.    3.
 min_val  =
    1.
-->[max_val max_pos] = max(A)

 max_pos  =
    2.    3.
 max_val  =

    6.
```

mean()

- Уже упоминалась команда среднего значения.
- mean() высчитывает среднее значение всех элементов матрицы.
- mean (, 'R') возвращает вектор-строку, состоящую из среднего значения каждого столбца.
- mea(, 'C') возвращает вектор-столбец, содержащий среднее каждой строки



```
-->A=[1 2 3; 4 5 6]
 A  =
    1.    2.    3.

    4.    5.    6.

 -->mean(A)

 ans  =
    3.5

 ->mean(A, 'r')

 ans  =

     2.5    3.5    4.5
-->mean(A, 'c')

 ans  =
    2.
    5.
```

size()

- Функция size()может быть использована для нахождения размера матрицы
- Ответ дается как число строк и стобцов (в таком порядке)
- Когда называются переменные строк и столбцов, ответ дается в алфавитном порядке (сначала столбцы)
- Матрицы с элементами строк (строки были использованы в диалоговом окне в Упражнении 1-3 и будут обсуждаться в деталях позже) так же обрабатываются

```
-->v1 = [1 2 3 4];

-->v2 = v1';

-->size(v1)

ans  =

   1.    4.

-->size(v2)

ans  =

   4.    1.
```

```
-->A = [1 2 3 4; 5 6 7 8];

-->size(A)
ans  =

   2.    4.
```

```
-->[n,m] = size([1 2 3; 4 5 6])

m  =

   3.
n  =
```

```
-->size(['You' 'Me'; 'Alpha' 'Beta'; 'Two' 'Three'])

ans  =
   3.    2.
```

length()

- Функция length() связана с size(). Для матрицы с числовыми элементами length() выводит количество элементов
- Для матрицы со строкой элементов length() выводится количество символов в каждом элементе
- Обратите внимание, что матрицы с смешанными числовыми и строковыми элементами не допускаются

```
-->length([1.23; 456,7890; 9])

 ans  =
    3.
```

```
-->length(['Hello world' 'SCILAB'; 'Alpha' 'Beta'])

 ans  =

    11.    6.

    5.     4.
```

find(состояние)

- Функция find() определяет и выдает расположение строк of those matrix elements that satisfy the Boolean condition stated in the argument
- Пустая матрица ([]) выдается в случае, когда нет элемента, удовлетворяемого заданному условию
- В заявлении X=3, логическое состояние является недопустимым. Хотя числовой ответ выдается, он считается не верным
- Позже мы сможем найти тот find(), который также может быть использован со строками

```
-->X = [9 1 8; 2 7 3; 6 3 5];

-->find(X<5)
ans  =

    2.    4.    6.    8.

-->find(X==3)
ans  =

    6.    8.

-->find(X=3)
ans  =

    1.

-->find(X~=3)
ans  =

    1.    2.    3.    4.    5.    7.    9.
```

gsort()

- Scilab не признает функцию Matlab sort() (используется в версиях до 5.3). Вместо этого мы должны использовать gsort(), которая отличается, но служит той же цели
- Кк показано справа, gsort() выделяет матричные элементы в порядке убвания и выводит их столбец за столбцом
- Мы используем Matlab как сортировку, добавив аргументы 'r' (строка) и 'i' (увеличение) в gsort()
- Используйте Help для получения дополнительных сведений об аргументах

```
-->matr = [-1 4 -2 2; 1 0 -3 3; -4 5 0 -5]
matr  =

   - 1.    4.  - 2.    2.
     1.    0.  - 3.    3.
   - 4.    5.    0.  - 5.

-->s_matr = gsort(matr)
s_matr  =

     5.    2.    0.  - 3.
     4.    1.  - 1.  - 4.
     3.    0.  - 2.  - 5.
```

```
-->matr = [-1 4 -2 2; 1 0 -3 3; -4 5 0 -5];


-->Mtlb_sort = gsort(matr, 'r', 'i')
 Mtlb_sort  =

 - 4.    0.  - 3.    -5.
 - 1.    4.  - 2.     2.
   1.    5.     0.    3.
```

testmatrix()

- Магические квадраты разных размеров могут быть заданы функцией testmatrix('magi',n). Это тоже самое, что и функция magic(n) в Matlab
- Дополнительные матрицы, которые можно получить с помощью функции testmatrix() является testmatrix('frk',n), которая выдает матрицу Френка и testmatrix('hilb',n), которая является обратной по отношению к nxn матрицы Гилберта. Используйте Help для получения деталей

```
-->testmatrix('magi',4)
 ans  =

    16.    2.    3.    13.
     5.    11.   10.    8.
     9.    7.    6.    12.
     4.    14.   15.    1.

-->testmatrix('magi',5)
 ans  =

    17.   24.   1.    8.    15.
    23.   5.    7.    14.   16.
     4.   6.    13.   20.   22.
    10.   12.   19.   21.   3.
    11.   18.   25.   2.    9.
```

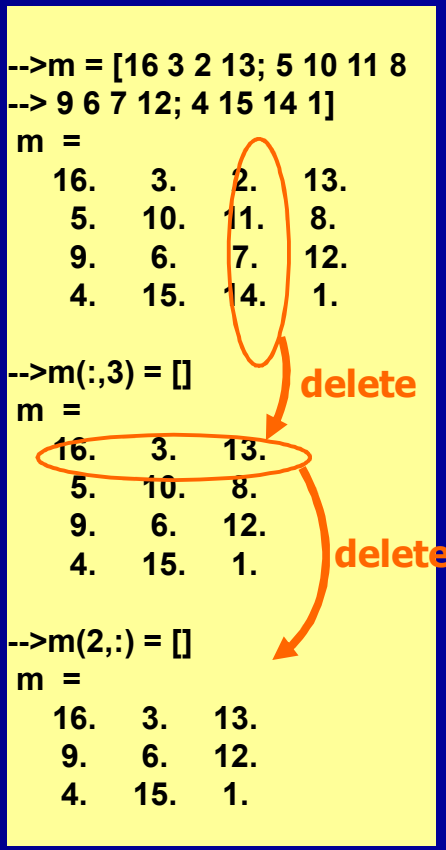
det(M) и ошибки округления

- Практические проблемы часто требуют рассчитать определитель (квадрат) матрицы
- Команда `det()` выдает определитель
- Определитель магического квадрата Дюрера равен нулю (матрица вырождена), но,как показано, ошибка округления предотвращает Scilab от вывода точного ответа (напомним, что мы сталкивались с этой проблемой раньше)
- Чтобы избавиться от ошибки округления, можно использовать функцию `clean()`. Это выдает ноль при значениях ниже 1e-10

```
-->M = testmatrix('magi',4)
 M  =

   16.    2.    3.    13.
   5.    11.    10.    8.
   9.    7.    6.    12.
   4.    14.    15.    1.

-->det(M)
 ans  =

 - 1.450D-12
```

Удаление строк и столбцов

- Строки и столбцы могут быть удалены с помощью пары квадратных скобок
- Начнем с матричного квадрата 4x4, обозначенного “m”, так как нам следует искажать его
- Сначала мы удалим третью колонку. Аргумент оператора столбцов используется для сохранения всех строк, аргумент 3 точки в третьем столбце. Результатом является матрица 3x4
- Во втором случае мы удаляем вторую строку, чтобы закончить с матрицей 3x3



Изменение строк и столбцов

- Логика предыдущего слайда может быть использована для изменяющихся строк и столбцов
- Начнем с предыдущей матрицы 3x3
- Сначала изменим элементы во втором ряду на нули
- Так же мы изменим элементы последнего столбца на единицы (транспонирование)
- Эти операции так же можно рассматривать как вставки определенного вектора строки/столбца вместо существующей строки/столбца

```
m  =
    16.     3.    13.
     9.     6.    12.
     4.    15.     1.

-->m(2,:)=[0 0 0]
m  =
    16.     3.    13.
     0.     0.     0.
     4.    15.     1.

-->m(:,3)=[1 1 1]'
m  =
    16.     3.     1.
     0.     0.     1.
     4.    15.     1.
```

Обращение матричных элементов путем линейной индексации

```
-->M=testmatrix('magi',4)

M  =
   16.    2.    3.    13.
    5.   11.   10.     8.
    9.    7.    6.    12.
    4.   14.   15.     1.
```

```
-->M(14)
ans  =
   8.
```

```
-->M([1  6  11  16])

ans  =
   16.
   11.
   6.
   1.
```

- Scilab касается матрицы, как векторов-столбцов. Это позволяет нам решать матричные элементы в упрощенном виде
- Мы начнем с магического квадрата 4x4
- Затем мы выберем элемент (2,4), что число 14, если считать по столбцам
- Следующее, выберем элементы главной диагонали
- Наконец, изменим элементы второй диагонали на нули

```
-->M([4 7 10 13]) = [0 0 0 0]

M  =
   16.    2.    3.    0.
    5.   11.    0.    8.
    9.    0.    6.   12.
    0.   14.   15.    1.
```

Объединение (1/2)

- Присоединением является процесс присоединения небольших матриц, чтобы сделать большую одну
- На самом деле, даже самая простая матрица формируется путем объединения отдельных ее элементов
- Пара квадратных скобок [] является оператором конкатенции
- Примеры иллюстрируют два основных случая объединения (с той лишь разницей, что во втором случае транспонированная матрица)
- Заметим, что если точка с запятой (;) ставится после команды - результат подавляется, но с запятой (,), он отображается (случай сверху)

```
-->A = [1 2 3]; B = [4 5 6], C = [A,B]

B  =
   4.    5.    6.
C  =
   1.    2.    3.    4.    5.    6.
```

```
-->A = [1 2 3]'; B = [4 5 6]'; C = [A,B]

C  =
   1.    4.

   2.    5.

   3.    6.
```

Объединение (2/2)

- В этом примере матрица 4x4 была создана путем объединения четырех матриц 2x2
- Линии были наложены для выделения слитых частей
- С другой стороны, мы могли бы объединить четыре строки или столбца векторов, матрица 3x3 плюс строка и столбец вектор и т.д.
- E может быть обработана к обычная матрица 4x4. Например команда `A = E(2:3, 2:3)` выделяет подматрицу

$$\begin{bmatrix} 22 & 23 \\ 32 & 33 \end{bmatrix}$$

```
-->A = [11 12; 21 22];

-->B = [13 14; 23 24];

-->C = [31 32; 41 42];

-->D = [33 34; 43 44];

-->E = [A B; C D]
 E  =

   11.   12.   13.   14.
   21.   22.   23.   24.
   31.   32.   33.   34.
   41.   42.   43.   44.
```

Операторы (1/4): Оператор столбца(:)

- Оператор столбцов (:), появился в предыдущих слайдах
- Это один из самых важных операторов в Scilab
- Обычно используется в виде:

```
0:%pi/36:%pi
```

  Значение: “начиная с 0, шаг π/36 до π”
- Первый пример показывает, что укороченная форма 1:8 производит вектор строку с приращением к 1. Вторая показывает, как обратиться к строками 3-4, столбец 2, из магического квадрата

```
-->1:8

ans  =

   1.   2.   3.   4.   5.   6.   7.   8.
```

```
-->M = testmatrix('magi',4)

M  =

   16.    2.    3.    13.
   5.     11.   10.   8.
   9.     7.    6.    12.
   4.     14.   15.   1.

-->K = M(3:4,2)

K  =
   7.

   14.
```

Операторы (2/4): Больше примеров с (:)

- Второй пример на предыдущем слайде был случаем подсценарной манипуляции типа M(i:j,k), где i:j относятся к i:th по j:th строкам и k к k:th столбцам
- Часто существуе наобходимость решать часть матрицы. Идея должна быть довольно понятна. Ниже приведены еще три примера
- Обратите внимание, что оператор столбцов самостоятельно относится ко всей строке или всему столбцу

```
-->M = testmatrix('magi',4);

-->A = M(2:3,2:3)

A  =

   11.    10.

   7.     6.
```

```
-->M = testmatrix('magi',4);

-->B = M(:,3)

B  =
   3.
   10.
   6.
   15.
```

```
-->M = testmatrix('magi',4);

-->C = M(3:4,:)

C  =

   9.    7.    6.    12.

   4.    14.   15.   1.
```

Операторы (3/4): Оператор $

- Оператор $ относится к последнему значению, $-1 к значению рядом с последним и т.д.
- Пример справа показывает некоторые способы использования оператора $
- Оператор $ может использоватся, чтобы перевернуть порядок элементов в векторе, как показано ниже (альтернативный метод был продемонстрирован в примере 1-3, задание 1)

```
-->M = testmatrix('magi',4)
 M  =

    16.    2.    3.    13.
    5.    11.    10.    8.
    9.    7.    6.    12.
    4.    14.    15.    1.

-->M($)
 ans  =
    1.

-->M(1:$-1,$)
 ans  =

    13.
    8.
    12.
```

```
-->v = [3 4 5 6 7 8 9];
->v($:-1:1)

 ans  =
    9.    8.    7.    6.    5.    4.    3.
```

Операторы (4/4): оператор обратный слеш (\)

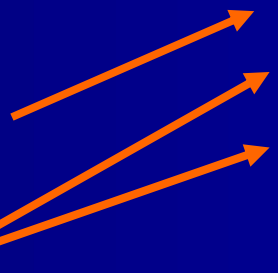
- Обратный слеш (\) обозначает деление левой матрицы. `x=A\b` является решением для `A*x=b` , что важно, например, в контрольнойтехнике
- Если A квадратная и невырожденная, `x=A\b` эквивалентно `x=inv(A)*b`, но время вычисления меньше, а результат является более точным.
- Здесь вы можетеувидеть дающиеся предупреждения, когда Scilab видит особенность в левом делении

- В этом случае Matlab производит ответы, отличные от тех, что дает Scilab

```
-->A = [3 -9 8; 2 -3 7; 1 -6 1]; b = [2 -1 3]';

 -->x = A\b

Warning :

matrix is close to singular or badly
scaled. rcond =    4.1895D-18

computing least squares solution. (see
lsq).

 x  =

    0.

 - 0.5641026

 - 0.3846154
```

Дублирование вектора mx1 к матрице mxn

- Оператор столбец позволяет дублировать векторы для формирования матрицы.
- Предположим, что у нас есть вектор-столбец m = (2:2:6)’; это означает, что он имеет три ряда
- Мы хотим сформировать матрицу 3x4, где каждый столбец состоит из вектора m



```
-->m = (2:2:6)';

-->n = 4;

-->A = m(:, ones(n,1))
A  =

    2.    2.    2.    2.
    4.    4.    4.    4.
    6.    6.    6.    6.
```

Обратите внимание на команду `m(:, ones(n,1))`. На словах это может быть интерпретировано как: “образуют матрицу с колличеством строк, определенных вектором столбцом m и число столбцов определяется переменной n. Заполнить матрицу с единицами и умножить каждую строку на соответствующее значение m. Повторить только один раз.”

Сингулярность и левое деление

- Термины "особый" и "не особый" появлялись на предыдущих слайдах
- Требование неособых квадратных матрицявляется то, что определитель отличен от нуля. Рассмотрим следующие случаи:

$$\begin{vmatrix} 6 & 2 \\ 5 & 3 \end{vmatrix} = 6 \cdot 3 - 2 \cdot 5 = 8 \text{ , поэтому не особый}$$

$$\begin{vmatrix} 6 & 3 \\ 2 & 1 \end{vmatrix} = 6 \cdot 1 - 3 \cdot 2 = 0 \text{ , означает, что это особый}$$

- Ранее мы обсуждали, что магический квадрат Дюрера является особым, так как например матрица A = [1 2 3; 4 5 6; 7 8 9]
- Перед выполнением левого деления с квадратными матрицами, следует проверить, что определитель матрицы коэффициентов отличен от нуля, например путем тестирования `clean(det(A)) ~= 0`

Строки (1/6): это тоже матрицы

- Характер (письма, текст, специальные характеристики) строки может быть создан с помощью одинарных или двойных кавычек:

```
'This is a &#ck2 string', "and so is this"
```

- Типичное использование строк в сюжетных командах - определить название x и y-знаков. Другие виды использования интерактивного ввода и вывода (input(), disp(), etc.), и написание команд (write(%io(2),.....))
- Строки считаются как матрицы 1x1 (скаляр) в Scilab, но смешанный характер/численные цепочки, как правило, матрицы 1x3. Это показано на следующем слайде с помощью команды

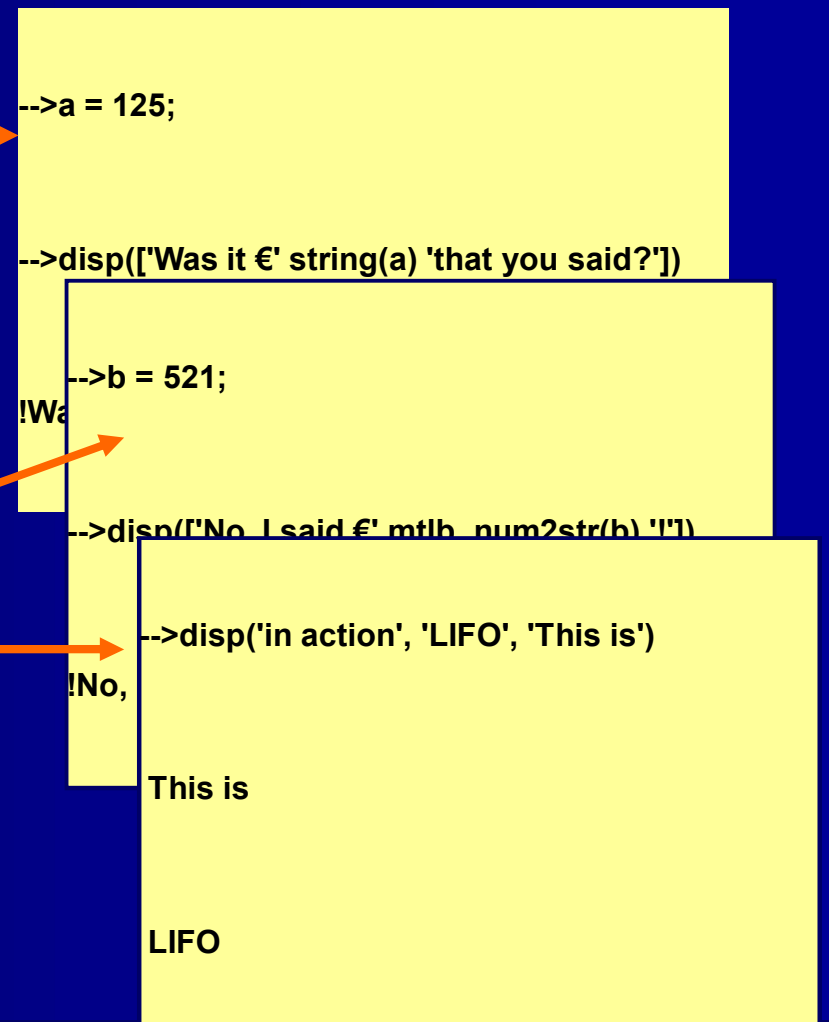
```
disp(['Was it €' string(a) 'that you said?'])
```

Элементы: 1 2 3

- Пример 2-4 показывает дополнительное приложение строк

Строки (2/6): disp(), string()

- Самая обычная комадна на дисплее это команда `disp()`, где текст должен быть в кавычках: `disp('text')` or `disp(['text'])`
- Числовые данные могут быть добавлены к `disp([])`, но должны быть преобразованы в строки с помощью функции `string()`
- Scilab знает преобразования командой из Matlab `num2str()`, но в виде `mtlb_num2str()`
- Уберите квадратные скобки и элементы отобразятся в видестолбца, начиная с последнего (Last In First Out)
- Запятые являются необязательными в квадратных скобках, но только не с простыми



Строки (3/6): disp() vs. mprintf()

- Как видно на предыдущем слайде, `disp()` дает вывод LIFO с пустой строкой между элементами
- Чтобы не появилась пустая строка, мы можем использовать функцию `mprintf()` с декларированной линией `\n` . В этом случае на выводе будет First In First Out. Обратите внимание, что аргумент является одной строкой
- Проверьте с помощью браузера для других приложений `mprintf()`

```
-->disp('in action', 'LIFO', 'This is')

 This is

 LIFO

-->mprintf('\nThis is \nFIFO \nin action')

This is

FIFO

in action
```

Строки (4/6): write(), input()

- Строчные аргументы в функциях `write()` и `input()` позволяют создавать интерактивные коды
- В показанном примере `write()` сначала используется, чтобы дать общую информацию для пользователя, после чего `input()` запрашивает данные необходимые при вычислении
- Аргумент `%io(2)` функции `write()` говорит, чтоцелью является консоль. Все действия после запущенного сценария в Scilab занимают место на консоли

```
// strings.sce                                    /
// Demo of write() and input() functions  /

clear,clc;
write(%io(2),'This is an interactive demo.');
write(%io(2),'You will be asked to give the base length'
write(%io(2),'and height of a triangle. Scilab then');
write(%io(2),'computes the area.');
write(%io(2),' ');        // Empty row
b = input('Give length of triangle base: ');
h = input('Give height of triangle: ');
write(%io(2),' ');        // Empty row
disp(['triangle_area = ' string(b*h/2)])
```

```
 This is an interactive demo.

You will be asked to give the base length

and height of a triangle. Scilab then

computes the area.
Give length of triangle base: 5

Give height of triangle: 4
!triangle_area =   10  !
```

Строки(5/6): другие полезные команды

Некоторые из функций, рассмотренных ранее в этой главе могут иметь строчные матрицы (ниже S) в качестве аргументов:

prod(), min(), max(), mean()	Не определено для строк
size(S)	Возвращает количество строк и столбцов в S
length(S)	Возвращает количество символов в каждой строке элемента
find(condition)	Возвращает размещение столбцам строкового элемента в матрице *
gsort(S)	Возвращает S с элементами перестроенными столбец за столбцом в алфавитном порядке убывания *

***) Смотрите демо-версию в следующем слайде**

Строки(6/6): демо-версия с find() & gsort()

- Справа мартица 3x2 называется “cars”
- Функция find() определяет и возвращает расположение указанной строки в матрице
- В случае, если совпадения нет, возвращается пустая матрица
- Функция sort() задает струнные элементы столбца за столбцом в алфавитном порядке убывания

```
-->cars = ['Audi' 'BMW' 'Fiat'; '343' 'Saab' 'Xantia']
cars  =

   !Audi  BMW   Fiat   !
   !                   !
   !343   Saab  Xantia  !

-->find(cars=='Saab')
ans  =

   4.

-->find(cars=='Volvo')
ans  =

   []

-->gsort(cars)
ans  =

   !Xantia   Fiat   Audi  !
   !                      !
   !Saab   BMW   343   !
```

Символьные вычисления

- Матрицы символьных строк строятся как обычные матрицы, например используя квадратные скобки
- Очень важной особенностью матриц символьных строк является способность манипулировать и создавать функции
- Символические манипуляции математических объектов можно выполнить с помощью матрицы символьных строк
- В представленных случаях функция `trianfml()` выполняет символьную трангуларизацию матрицы `sc`, и функция `evstr()` вычисляет выражение `tsc`

```
-->sc = ['x' 'y'; 'z' 'v+w']
sc  =

! x    y      !
!              !
! z    v+w    !

-->tsc = trianfml(sc)
tsc  =

! z    v+w            !
!                     !
! 0    z*y-x*(v+w)    !

-->x=1; y=2; z=3; v=5; w=4;

-->evstr(tsc)
ans  =

    3.    9.
    0.  - 3.
```

Массивы: общее

- Термин "матрица" относится к любому систематичному размещению объектов, обычно в виде строк и столбцов (числовые массивы, диодные линейки, антенные решетки, и т.д.)
- Массивы имеют некоторые важные области применения, например, для построения таблицы
- Арифметические операции над массивами выполняются элемент-на-элемента, что означает, что сложение и вычитание одинаковы для массивов и матриц
- Scilab использует оператор точка (.) Для операций над массивами

+	дополнение
-	вычитание
.*	умножение
./	правое деление
.\	левое деление
.^	сила
.'	Неконъюгированное транспонирование массива

Массивы: построение таблиц

- Предположим, что у нас есть вектор-столбец `n=(0 9)'`
- Затем мы можем построить таблицу с простой функции-в показанном случае с колоннами для `n`, `n^2` и `2^n`
- Этот тип таблиц полезен, например, при обработке данных измерений
- Второй пример показывает, что Scilab рассматривает созданную таблицу как нормальную матрицу

```
-->n = (0:9)';

-->powers = [n  n.^2  2.^n]
powers  =

    0.    0.    1.
    1.    1.    2.
    2.    4.    4.
    3.    9.    8.
    4.    16.   16.
    5.    25.   32.
    6.    36.   64.
    7.    49.   128.
    8.    64.   256.
    9.    81.   512.
```

```
-->p = powers(4:5,1:2)
p  =

    3.    9.
    4.    16.

-->q = powers(3,2)*powers(4,3)
q  =

    32.
```

Умножение и деление элемент-на-элемент

- Элемент-на-элемент умножение с использованием оператора точка (.) также может быть выполнена на двумерных матриц
- В первом примере мы умножим, элемент-на-элемент, две матрицы 2x2 для формирования матрицы 2x2 Продукт C
- Обратите внимание на иной результат с обычным матричным умножением
- И вот мы делим эти же матрицы элемент-на-элемент, чтобы сформировать матрицу 2x2 частных

```
-->A = [1 2; 3 4]; B = [5 6; 7 8]; C = A.*B
 C  =

    5.    12.
    21.   32.

-->D = A*B
 D  =

    19.   22.
    43.   50.

-->E = A./B
 E  =

    0.2         0.3333333
    0.4285714   0.5
```

Правое и левое деление

- Как показано в таблице выше, Scilab позволяет левое и правое деление элемент-на-элемент (. \ и . / соответственно)
- Разница между ними состоит в том, какой из двух элементов деления является числителем и который знаменателем
- Как показано на примерах, слева разделение означает, что элемент в левой матрице становится знаменателем, с правым разделением - числителем

Показательная функция exp () является частным случаем, в настоящее время определяется как элемент-за-элементом работы

```
-->A = [1 2; 3 4]
 A  =

    1.    2.
    3.    4.

-->B = [5 6; 2 -3]
 B  =

    5.    6.
    2.  - 3.

-->A.\B
 ans  =

    5.                3.
    0.6666667   - 0.75

-->A./B
 ans  =

    0.2      0.3333333
    1.5   - 1.3333333
```

Ловушка оператора точки (.)

- В практическом моделировании Scilab часто мигает сообщения об ошибках из-за неправильного использования оператора точки,или его отсутствия
- Особой проблемой является разделение с целым числом в числителе. Как показано здесь, первый случай интерпретируется Scilab как B = (1.0)/A и второй как C = (1.0)./A Постарайтесь запомнить!
- Тот,кто имеет опыт работы с Matlab должны знать, что приоритет оператора точки отличается в Scilab
- Это не проблема с умножением

```
-->A = [1 2 3 4];

-->B = 1./A
 B  =

    0.0333333
    0.0666667
    0.1
    0.1333333

-->C = (1)./A
 C  =

   1.    0.5    0.3333333    0.25

-->D = 2.*A
 D  =

    2.    4.    6.    8.

-->E = (2).*A
 E  =

    2.    4.    6.    8.
```

Еще несколько функций (1/5): modulo()

- Тогда команда modulo(n,m) вычисляет остаток при делении n на m (где n и m целые числа)
- С матрицами modulo() вычисляет остаток элемент-на-элемента
- modulo() попадется под руку, например, когда мы должны проверить, является ли число четным или нечетным (if-then-else-end конструкции будут обсуждаться в главе 11)
- Существует также функция pmodulo(). Проверьте в помощи

```
-->modulo(3,2)

ans  =


  1.
```

```
-->n=[1,2; 10,15]; m=[2,2; 3,5];

-->modulo(n,m)

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

```
x=input('Give a number :');
if modulo(x,2)==0 then
    disp('Number is even');
else
    disp('Number is odd');
end
```

```
Give a number :24

 Number is even
```

```
Give a number :1443
 Number is odd
```

Еще несколько функций (2/5): getdate()

- Мы видели getdate() в действии уже в примере 1-3, где он был использован для улучшения случайности
- Другое использование getdate() является постановкой штампа с датой на распечатке моделирования
- getdate() имеет многочисленные альтернативные аргументы. В дополнение к тем, которые используются в примере 1-3 есть еще на примере справа.
- Начальной точкой "часов" от getdate() является UTC 00:00 on 1 января 1970

```
-->xp=getdate();

-->xp(1),xp(2),xp(3),xp(4),xp(5),xp(6),xp(7)
ans  =

   2011.          (1) present year
ans  =

   3.             (2) present month
ans  =

   12.            (3) present week
ans  =

   83.            (4) day of the year
ans  =

   5.             (5) weekday (Thu)
ans  =

   24.            (6) day of the month
ans  =

   11.            (7) hour of the day
```

Еще несколько функций (3/5): unique()

- Напомним что unique() был использован в Пимере 1-3 в состояни length(unique(numbers))<7 констатировать, что ряд лото содержал только уникальные номера
- Здесь unique() используется для идентификации, генерируемые целые числа в диапазоне [1,5]
- Во втором случае unique() выделяет уникальные строки в матрице. Поменять ‘r’ на ‘c’ чтобы найти уникальные столбцы
- Сравните unique() с find(), который обсуждался ранее

```
-->M=round(5*rand(5,1))'

 M  =

   5.   2.   1.   5.   4.
-->unique(M)
 ans  =
   1.   2.   4.   5.
```

round()
см.пример Ex. 1-3

0 и 3 отсутствуют

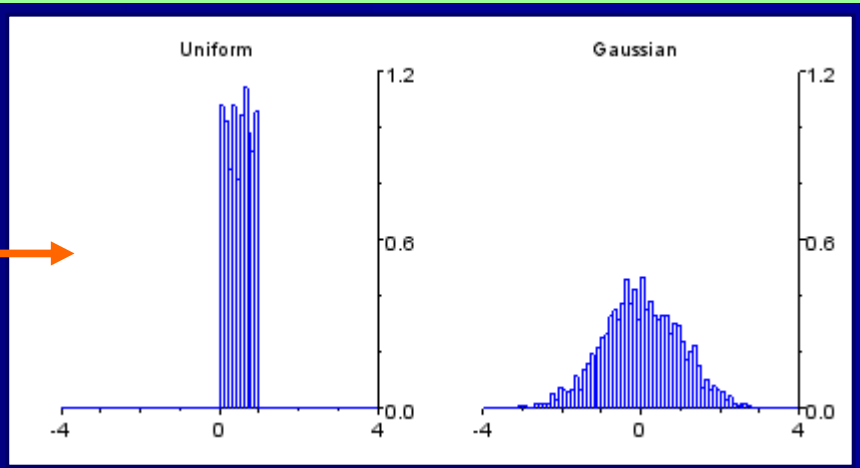
```
A = [0 0 1 1;
     0 1 1 1;
     2 0 1 1;
     0 2 2 2;
     2 0 1 1;
     0 0 1 1 ];

disp(['Unique rows are:'])
disp(unique(A,'r'))
```

```
Unique rows are:

   0.   0.   1.   1.

   0.   1.   1.   1.

   0.   2.   2.   2.

   2.   0.   1.   1.
```

Еще несколько функций (4/5): rand()

- Мы видели случайный числовой генератор rand() уже несколько раз
- rand() может генерировать два типа чисел, либо с равномерным или распределением Гаусса. Равномерное по умолчанию, распределение Гаусса (нормальное) выбирается с помощью аргумента ‘normal’ (или ‘n’)
- Справа находятся гистограммы 2000 случайных чисел, генерируемых с равномерным и распределением Гаусса (последние со средним 0, дисперсия 1)

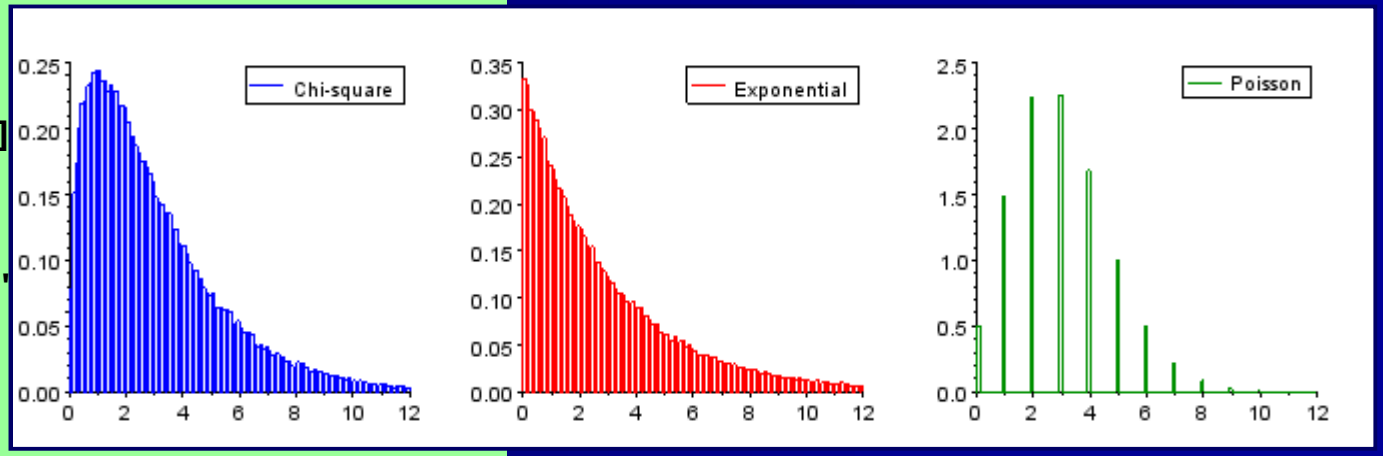
```
// rand_demo1.sce
clear,clc,clf;
u_fun=rand(1,2000);
subplot(121);
histplot([-4:0.1:4],u_fun,2,'073',' ',[-
4,0,4,1.2],[3,3,2,3]);
xtitle('Uniform')

G_fun=rand(1,2000,'n');
subplot(122);
histplot([-4:0.1:4],G_fun,2,'073',' ',[-
4,0,4,1.2],[3,3,2,3]);
xtitle('Gaussian')
```



Еще несколько функций (5/5): grand()

```
// grand_demo.sce

// Plot histograms of Chi-square, exponential, /
// and Poisson distributions with 10^5 draws /

clear,clc,clf;

Chi=grand(100,1000,'chi',3)   // Chi, 3 deg freedom
Exp=grand(100,1000,'exp',3) // Exponential, mean 3
Poi=grand(100,1000,'poi',3)   // Poisson, mean 3

x = [0:.1:12];
subplot(131);
histplot(x,Chi,2)
legend(['Chi-square']

subplot(132);
histplot(x,Exp,5)
legend(['Exponential'

subplot(133);
histplot(x,Poi,13)
legend(['Poisson'])
```

Фнкция grand() является более универсальной чем rand(). Это позволяет генерировать большинство существующих дистрибутивов. На рисунке an вы видите пример с Chi-квадратом, показатели и распределения гистограмм Пуассона



Dr. EW
Johnny Heikell

6. Примеры, набор 2

Добавление к тому, что мы узнали ранее

Пример 2-1: решение системы уравнений

- Задача состоит в том, чтобы решить следующую систему уравнений:

$$\begin{aligned} x_1 + 2x_2 - x_3 &= 1 \\ -2x_1 - 6x_2 + 4x_3 &= -2 \\ -x_1 - 3x_2 + 3x_3 &= 1 \end{aligned}$$

- Мы можем записать его в матричном виде Ax = b, где:

$$A = \begin{bmatrix} 1 & 2 & -1 \\ -2 & -6 & 4 \\ -1 & -3 & 3 \end{bmatrix}, \quad b = \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix}$$

- Далее настроим уравнения в Scilab, чтобы найти решение x:

Пример 2-1: сценарий и решение

Код для системы уравнений вводится и называется algebra1.sce. Обратите внимание на оператор обратного слеша (\)

```
// algebra1.sce

// Find the solution to x in  /
// Ax = b                     /

A = [1 2 -1; -2 -6 4; -1 -3 3];
b = [1; -2; 1];
x = A\b
```

algebra1.sce должен быть запущен с консоли, так как сценарий не содержит команду disp()

Решение:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -1 \\ 2 \\ 2 \end{bmatrix}$$

```
-->exec algebra1.sce

-->// algebra1.sce             /
-->//
-->// Find the solution x in   /
-->// Ax = b                   /
-->//

-->A = [1 2 -1; -2 -6 4; -1 -3 3];
-->b = [1; -2; 1];
-->x = A\b
 x  =

   - 1.
     2.
     2.
```

Пример 2-1: проверка результата

- Это хорошая практика, чтобы проверить свои решения
- В этом случае это может быть сделано, убедившись, что остаточное $B - Ax$ точно равно нулю
- Измененный код переименован algebra1_check.sce и спас перед выполнением
- В результате 0, как мы и надеялись (обратите внимание, что ошибки округления здесь нет; результат в точности равен нулю)

```
// algebra1.sce

// Find the solution x in  /
// Ax = b                  /

A = [1 2 -1; -2 -6 4; -1 -3 3];
b = [1; -2; 1];
x = A\b

// algebra1_check.sce          /
// Make sure that b - Ax = 0   /

residual = b - A*x
```

```
x  =


 - 1.

   2.

   2.


residual  =


   0.

   0.

   0.
```

Пример 2-1: что должно было быть сделано до этого

Проблема: Определитель матрицы коэффициентов A должен быть отличен от нуля

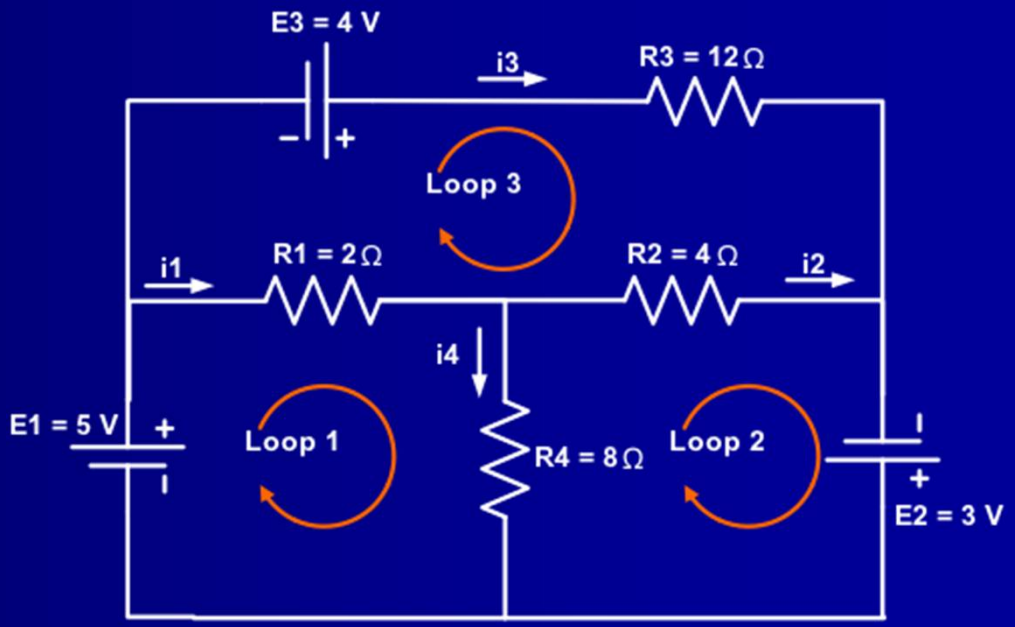
```
-->A = [1 2 -1; -2 -6 4; -1 -3 3];

-->det(A)
 ans  =

  - 2.
```

- В соответствии с тем, что было сказано ранее, мы должны начать с проверки, что определитель матрицы коэффициентов A не особенный
- Мы можем проверить его в ретроспективе и увидеть, что это не так
- При написании программы для практического применения, мы должны включить нулевую проверку в сценарий. Это, однако, требует управления потоком (условного ветвления), что будет обсуждаться в Главе 11

Пример 2-2: решение токов в цепи постоянного тока

Задача: Определить четыре токи i1, i2, i3, i4 для показанной цепи постоянного тока

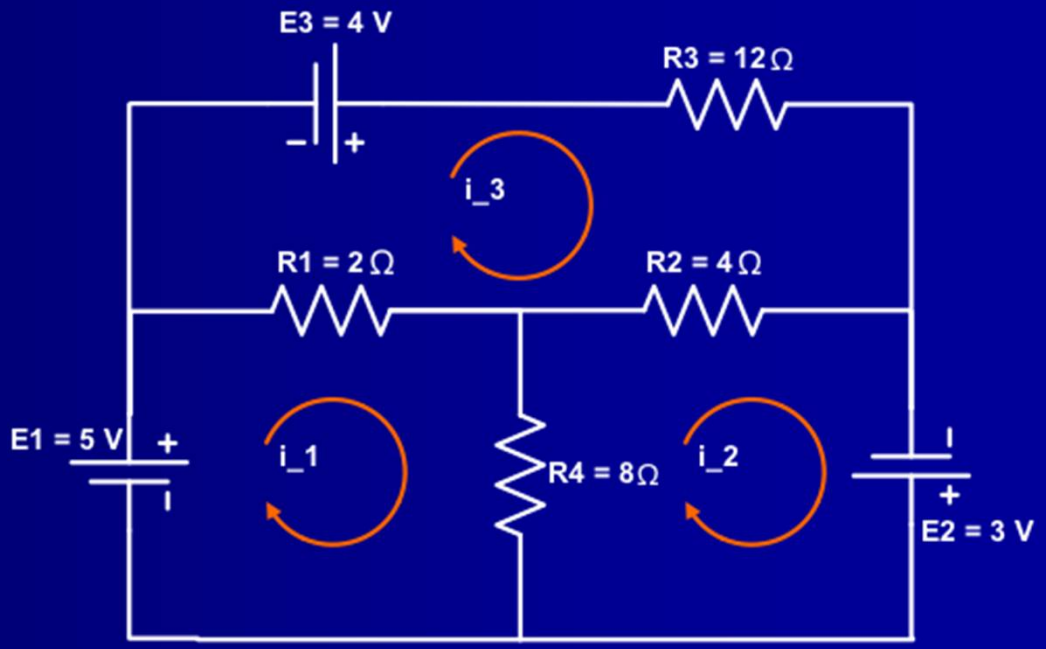
Фигура позволяет закон напряжения Киргофа который должен выполняться. Тем не менее, метод приводит к не квадратной матрицы и инструментов и оператор обратный слеш (\) и умножения с обратных матриц не может быть применен



Пример 2-2: сетка тока

Вместо этого, суперпозиция токов с сеткой текущих уравнений могут быть использованы. Вдоль токовой петли диагональные сопротивления получаются:

R11 = 10 Ω
R22 = 12 Ω
R33 = 18 Ω



Общие (внедиагональные) сопротивления являются: R12 = -8 Ω, R13 = -2 Ω, R21 = -8 Ω, R23 = -4 Ω, R31 = -2 Ω, R32 = -4 Ω (Вы должны быть в состоянии выяснить логику)

Пример 2-2: решение

Эти значения позволяют записать следующие уравнения сетки тока:

$$\begin{bmatrix} 10 & -8 & -2 \\ -8 & 12 & -4 \\ -2 & -4 & 18 \end{bmatrix} \begin{bmatrix} i\_1 \\ i\_2 \\ i\_3 \end{bmatrix} = \begin{bmatrix} 5 \\ 3 \\ 4 \end{bmatrix}$$

```
// circuit1.sce

// Mesh-current solution for Example
4 /

R = [10 -8 -2; -8 12 -4; -2 -4 18];
u = [5 3 4]';
i_n = R\u
residual = clean(u -  R*i_n)   // Check
```

```
i_n  =
   2.5
   2.25
   1.
residual  =

   0.
   0.
   0.
```

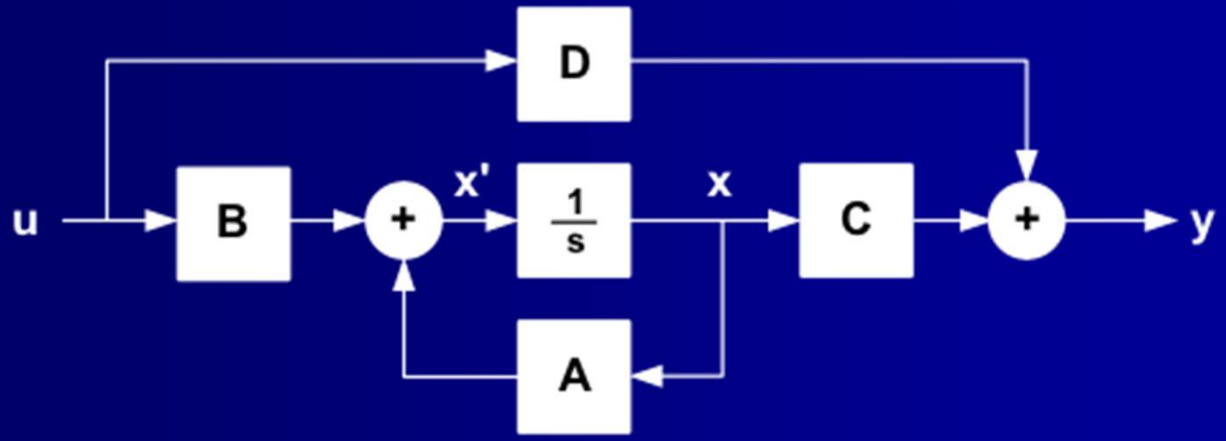
Мы запустим скрипт в консоли и вычислим вручную текущие значения, которые мы ищем:

```
i1 = i_1 - i_3 = 1.5 A
i2 = i_2 - i_3 = 1.25 A
i3 = i_3       = 1 A
i4 = i_1 - i_2 = 0.25 A
```

Пример 2-2: комментарии

- Пример показывает, что мы должны найти правильный метод, чтобы иметь возможность использовать матричные операции
- Есть ли причина для использования матриц, которые являются альтернативными?
- Первый вариант будет исходить из исходной диаграммы и применять закон напряжения Киргофа, и решить проблему вручную. Это довольно утомительное занятие
- Другой альтернативой является, чтобы начать расчеты вручную из набора сетки тока уравнений с помощью правила Крамера. Тем не менее, это также требует хорошую дозу алгебры, так как мы должны вычислить определитель
- Короче говоря, с помощью Scilab начинание манипулировать матрицами упрощается. С более сложными схемами разница еще более заметна

Пример 2-3: состояние модели в пространстве, непрерывном временем



- На рисунке показана типичное состояние модели в непрерывном временем пространстве, определяемое матричными уравнениями

$$x' = Ax + Bu$$
$$y = Cx + Du$$

Где
- A = матричная система
- B = ввод матрицы
- C = вывод матрицы
- D = упреждение матрицы
- x = состояние вектора
- x′ = dx/dt
- u = ввод вектора
- y = вывод вектора

Пример 2-3: задача

- Предположим, что система определяется по формуле:

$$A = \begin{bmatrix} 0 & 1 \\ -1 & -0.5 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 1 \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 0 \end{bmatrix}, \quad D = \begin{bmatrix} 0 \end{bmatrix}$$

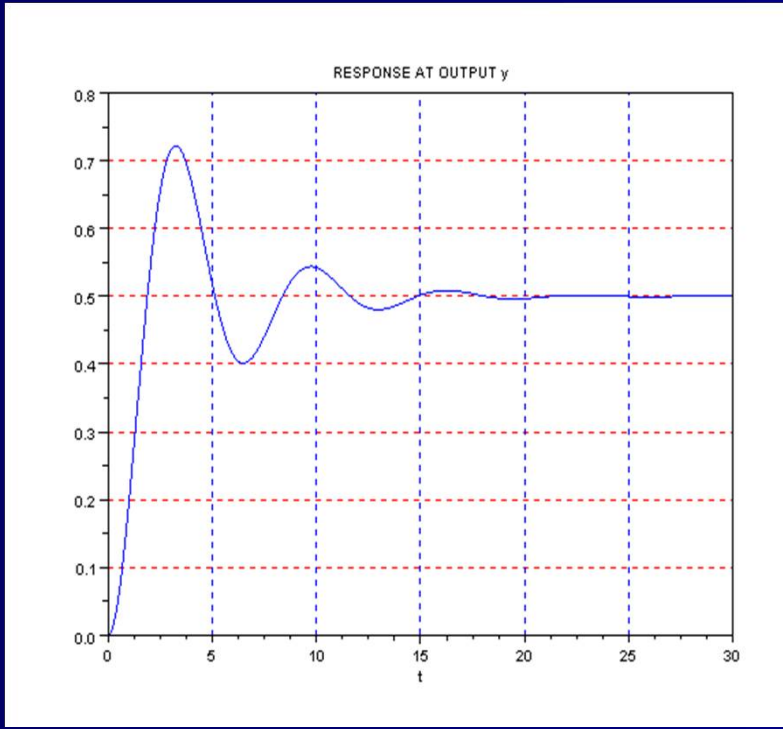
- Ввод постоянной u в 0.5
- Начальное состояние вектора $x0 = [0 \ 0]$, т.е., x = 0 в t = 0
- Задача состоит в том, чтобы построить вывод y и состояние переменных ответов (x, x’) для t = 0 … 30

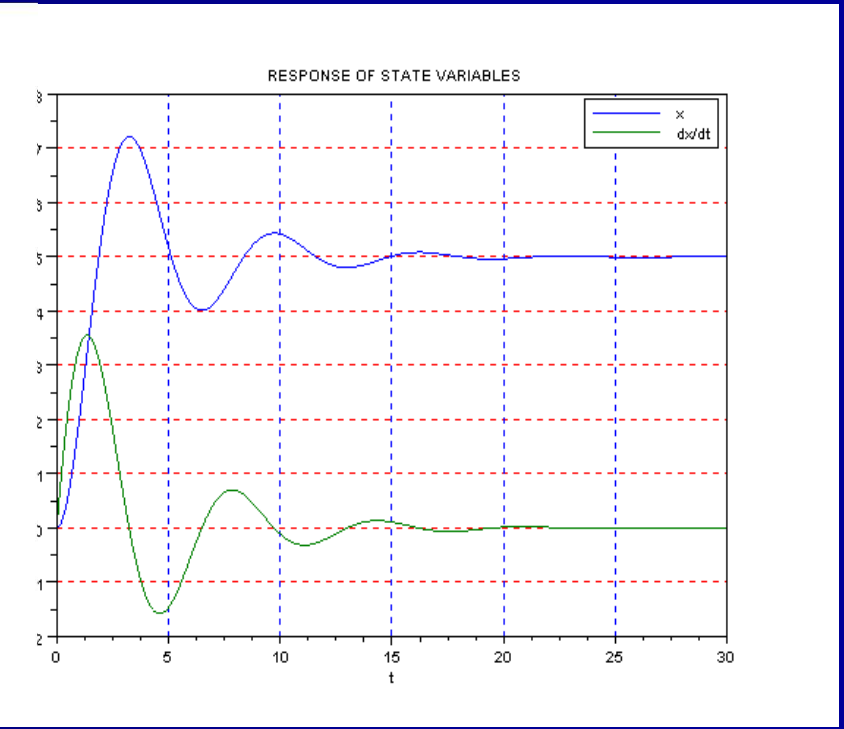
Пример 2-3: скрипт

- Впервые модель в пространстве определяется так
- Обратите внимение на функцию syslin() которая определяет линейную систему
- Затем ответы в связи с начальным состоянием и внешним входным сигналом y моделируются с помощью csim()
- Чтобы закончить, ответы при выводе y и состояние переменных x и x'are нанесены в отдельных окнах

```
// state_space.sce

// Simulates a continuous-time state-space   /
// system model                              /

clear,clc;
A=[0,1;-1,-0.5];              // System matrices
B=[0;1];
C=[1,0];
D=[0];
x0=[0;0];                     // Initial state
sys=syslin('c',A,B,C,D,x0);   // Create cont.-time ('c') system model
t=[0:0.1:30];                 // Time vector
u=0.5*ones(1,length(t));      // Create constant input signal

[y,x]=csim(u,t,sys);   // Compute with u=input, y=output, x=states

scf(1); clf;                          // Open and clear figure 1
plot(t,y);                            // Plot response in y
  xtitle('RESPONSE AT OUTPUT y','t');
  ax1=gca(); ax1.grid=[2,5];     // Handle: add grid to y-plot
scf(2); clf;                          // Open and clear figure 2
plot(t,x);                            // Plot response in x
  xtitle('RESPONSE OF STATE VARIABLES','t');
  legend('x','dx/dt',1);              // Add legend to x-plot
  ax1=gca(); ax1.grid=[2,5];    // Handle: add grid to x-plot
```

Пример 2-3: графики





Обратите внимение на использование функции `scf(номер)` (установить текущую фигуру) для построения двух графиков

Пример 2-3: комментарии (1/3)

- Помимо демонстрации матричных операций, этот пример ввел ряд новых понятий:
  - Определение линейной системы с функцией syslin (), в котором строка 'c' в качестве входного аргумента обозначает "непрерывно." начальное состояние x0 = [0; 0] не требуется, так как x0 = 0 значение по умолчанию, но это то, если мы хотим сделать изменения
  - В Scilab отсутствует блок ступенчатой функции; входной сигнал постоянной построен с единичным вектором (используя `ones()`) длиной = `t`
  - Моделирование определенной системы было сделано функцией `csim()`, с `u`, `t`, и `sys` в качестве входных аргументов
  - `csim()` производит аргументы вывода `y` и `x`, которые используются by при построении команд. Проверьте `Help` для подробного объяснения
  - Создаются два графика с определенной системой, `x` и `y` в противном случае перекрываются. `x` и `x'` строятся автоматически
  - `ax1=gca()` и `ax1.grid=[2,5]` пара команд говорит, что мы хотим сетку с синими вертикальными и красными горизонтальными линиями

Пример 2-3: комментарии (2/3)

Для линейной системы мы можем использовать либо функцию передачи или представление в пространстве. Их различия:

Функция передачи	Состояние в пространстве
Внешнее описание	Внутреннее описание
Описание ввода/вывода	Описание состояния
Метод частотного отклика	Метод временного отклика
Преобразование Лапласа	Матрица
Некоторые внутренние соединения скрыты	Показываются переменные состояния
Система становится более компктной с меньшим числом параметров	Представляется с несколькими параметрами
Еденичные ввод / вывод	Многочисленные ввод / вывод

В распространении являются блок схемы и их манипуляции, полюса и нули

Пример 2-3: комментарии (3/3)

- Можно переключаться между передаточными функциями и в состоянии пространства:
  - `tf2ss()`, Передаточная функция для состояния пространства
  - `ss2tf()`, Состояние пространства для передаточной функции
- Эти функции необходимы, например, когда дискретизации модели непрерывного времени, для которого Scilab имеет функцию DSCR (), но который действителен только для моделей состояния пространства
- См. учебник по Хаугену, раздел 9.6, для краткого обсуждения. Подробное обсуждение дается в устаревшей обработки сигналов с Scilab, разделы 1.5, 1.7, и 2.1 (вы можете получить доступ через <http://wiki.scilab.org/Tutorials>)

Пример 2-4: функции строки, скрипты

Этот пример относится к обсуждения о строках в главе 5 Chapter **5**

Строчные аргументы input()

Конструкции if...else...end будут обсуждаться в главе 11 Chapter 11

Обратите внимание на взаимодействие между floor() and modulo()

Строчные аргументы disp()

```
// conv_seconds.sce

// The script asks for a number of seconds,    /
// checks that the given number is positive,    /
// then converts the number into hours,        /
// minutes, and seconds                        /

clear,clc;
time = input("Give time in seconds: ");
if time < 0                                  // Check if time >= 0
  disp("ERROR, negative number")     // Display error
message
  abort                                  // and abort execution
else
  minut = floor(time/60);                // Convert to minutes
  seconds = modulo(time,60);          // Remaining seconds
  hours = floor(minut/60);               // Convert to hours
  minutes = modulo(minut,60);         // Remaining minutes
  disp(string(hours)+" hour(s) "...      // Display answer
  +string(minutes)+" minute(s) "...
  +string(seconds)+" second(s) ")
end
```

Пример 2-4: строковые функции, выполнение и комментарии

Ниже приведен результат из трех различных трасс

Give time in seconds: 0
0 hour(s) 0 minute(s) 0 second(s)

Give time in seconds: -3600
ERROR, negative number

Give time in seconds: 7465.33

2 hour(s) 4 minute(s) 25.33 second(s)

В сценарии, для начального очищенияиспльзуем команды `clear,clc;`. Если `clf` was включен, то это вызовет излишне всплывающее графическое окно (в примере 2-3 было бы произведено дополнительное окно)

В этом примере мы впервые использовали простой тест (если время <0 ...), чтобы убедиться, что у пользователя не вызывет проблем с неправильным вводом

В таком случае это раздражает, что консоль не становится активной после исполнения команды в редакторе. Вы автоматически начнете вводить в ответ, как только появляется командная строка, но курсор остается на редакторе ...

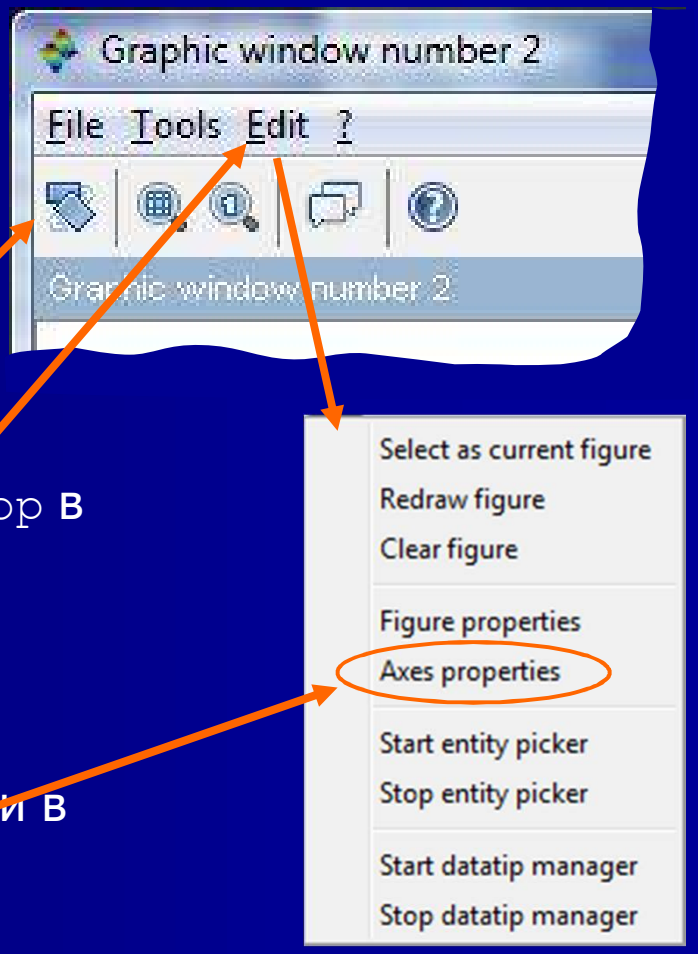
Dr. EW
Johnny Heikell

7. Графики

2D и 3D графики, сюжетные линии и другие типы графиков; редактирование графиков

Графические окна

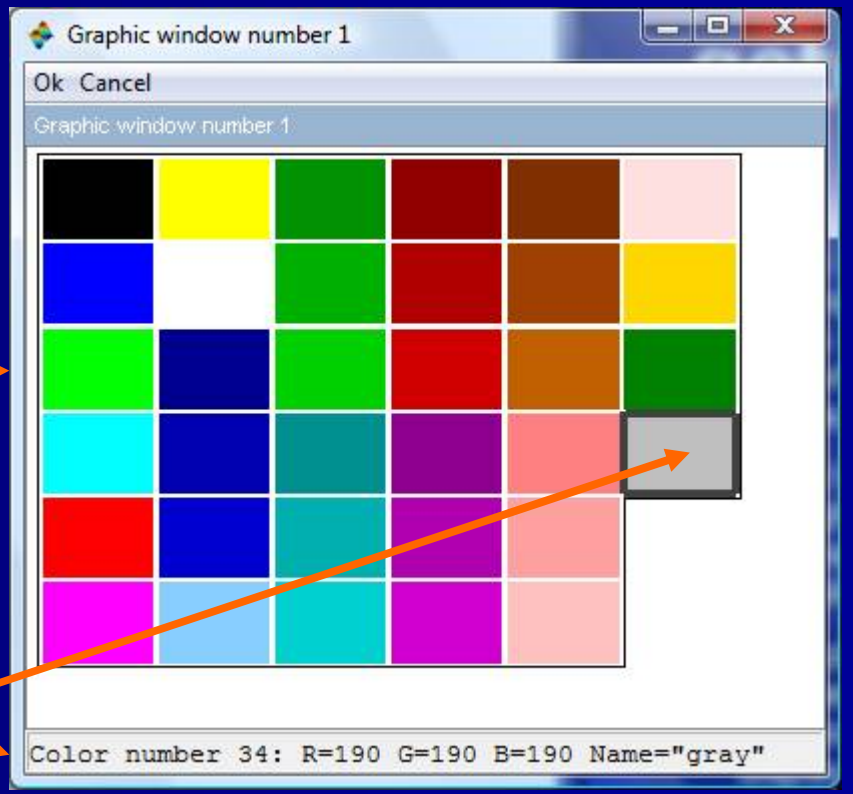
Важно: Функция Демонстрация имеет хорошую презентацию построения графика функции



- Панель инструментов позволяет вращение и зуммировани участка
- Действительно интересный `редактор` в строке меню, и Figure properties и Axes properties, что показаны при нажатии на Edit
- Однако, Figure properties не однозначная, все опции можно найти в разлеле Axes properties

getcolor()

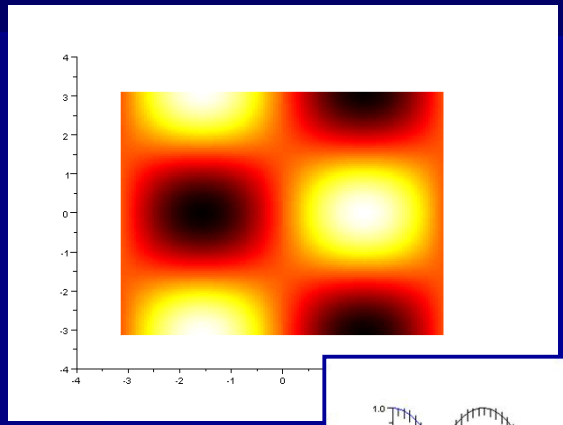
- При работе с графикой вы можете проверить, какие цвета доступны в Scilab и каковы их коды или названия
- Цветовая палитра Scilab может выводиться, введя команду `getcolor()` на консоли
- Нажав на нужный цвет в палитре его номер, гамма состав и имя отображаются в нижней части окна (Scilab 5.3.x не отображает два последних, 33 "зеленых" и 34 "серый"). *

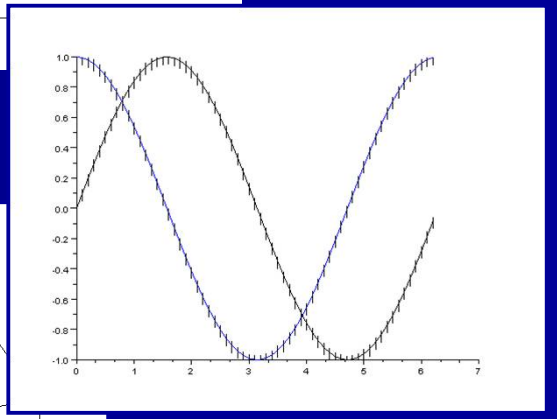


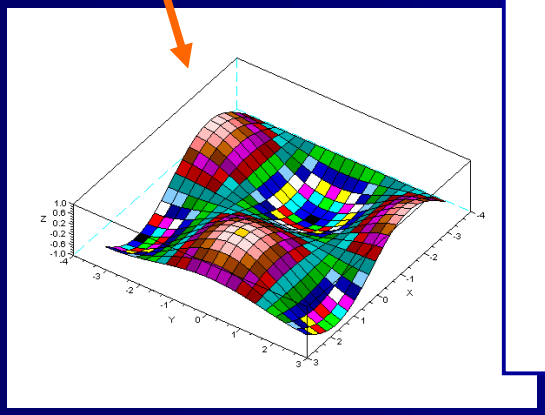
***) Я пропускаю более светлые тона для использования на заднем фоне графика.**

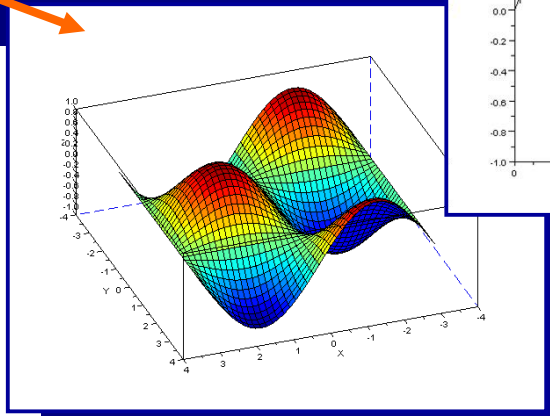
Демонстрация графических функций

- Вы получаете демо определенных графических функций введя имя функции на консоли. Примеры:
  - grayplot()
  - errbar()
  - plot3d()
  - fplot3d1()





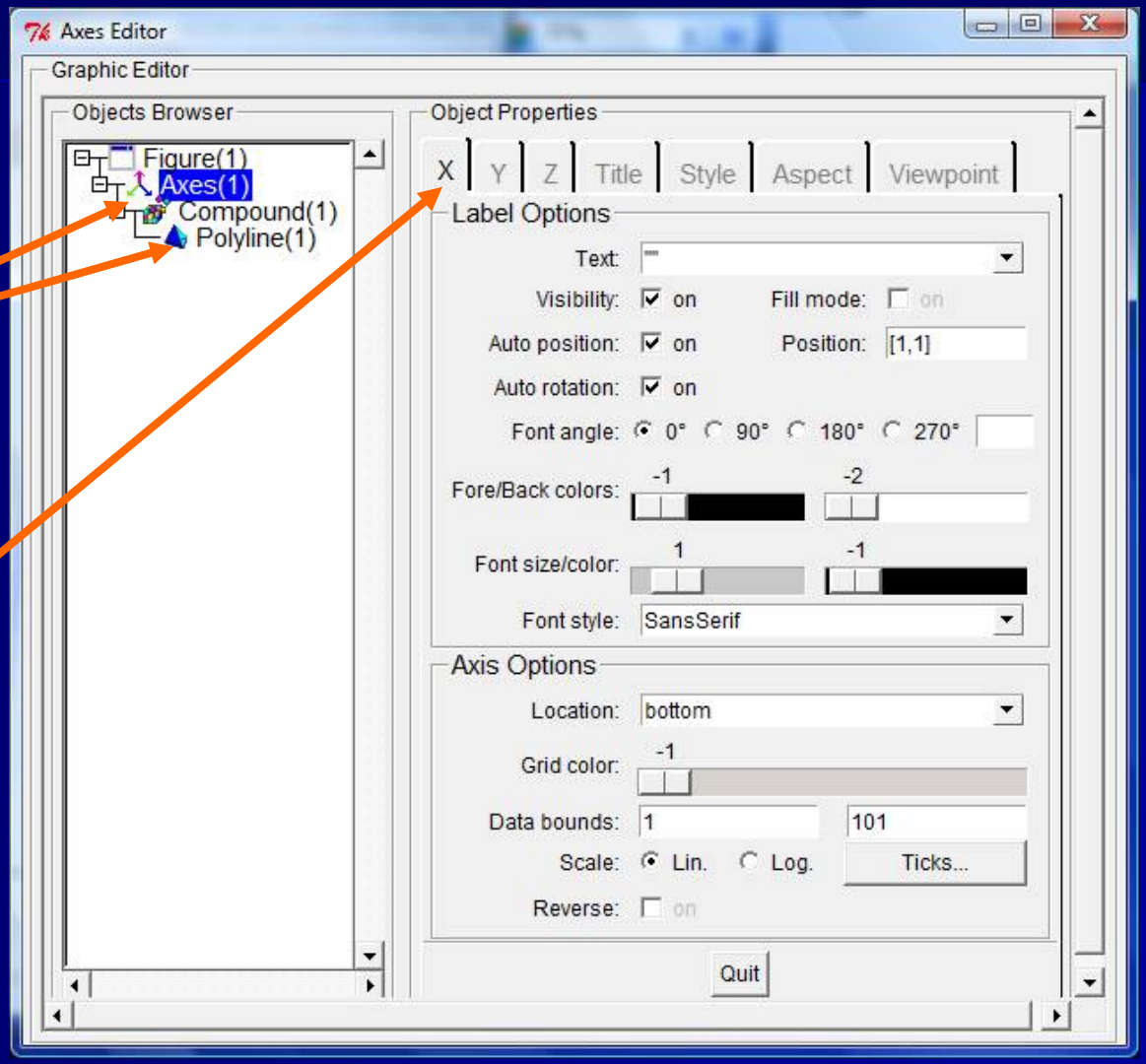




Редактор осей

Самые полезные объекты редактирования Axes() и Polyline(). Здесь открывается осей окно

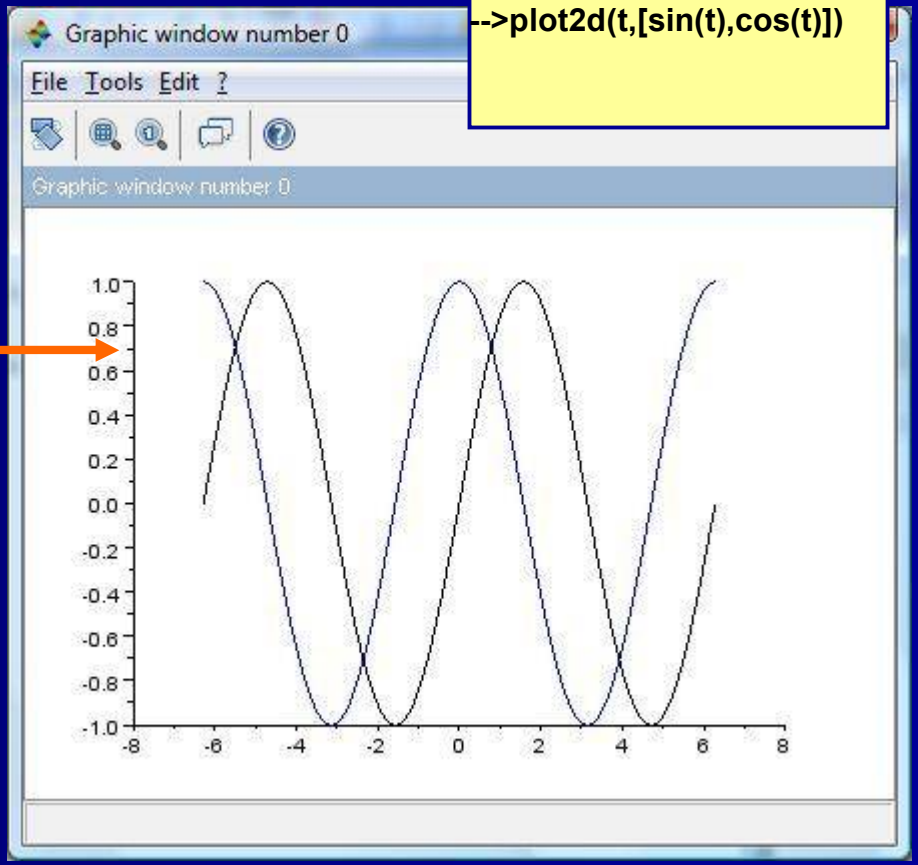
Есть семь свойств объекта, которые могут воспроизводиться, например с одним для x-оси как показано



Демонстрация редактирования графика, начальная точка

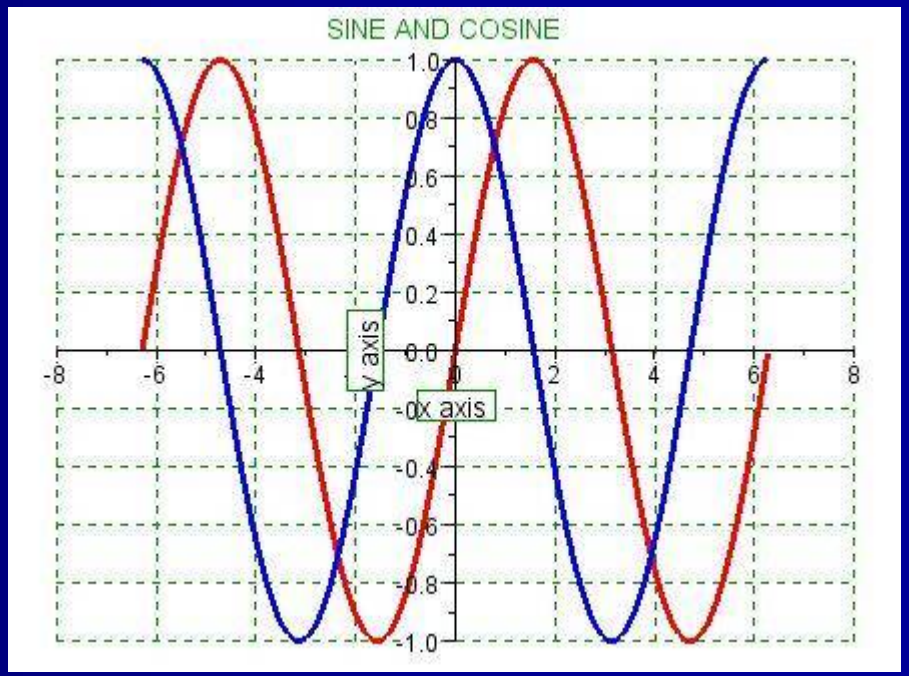
- Давайте начнем с построения кривых синуса и косинуса в одной плоскости
- В результате график не очень привлекательный
- Так давайте сделаем некоторые изменения, чтобы придать ему более привлекательный внешний вид
- Начнем с Edit\Axes properties

```
-->t=(-2*%pi:0.01:2*%pi)';

-->plot2d(t,[sin(t),cos(t)])
```



Демонстрация редактирования графика, Отрадектированный график

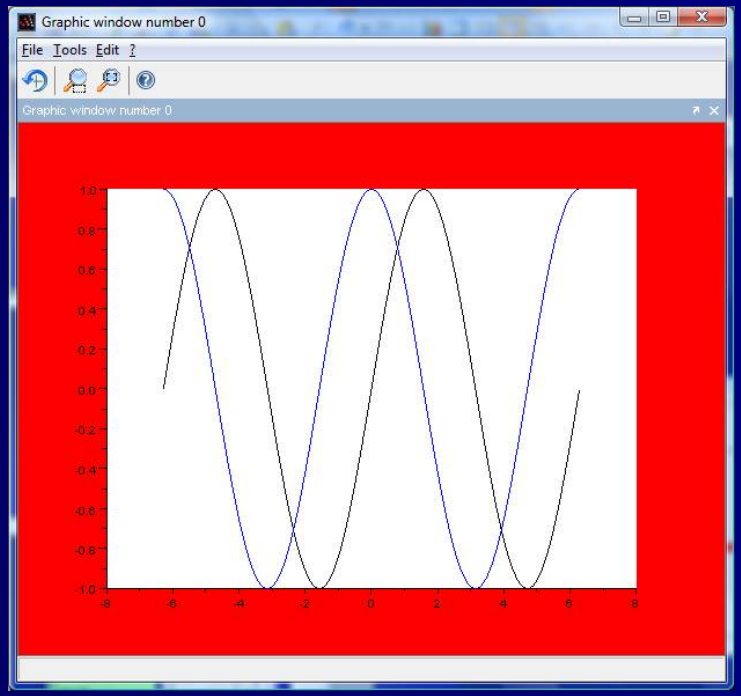
- Чтобы изменить цвета синуса и косинуса, выберите пункт: Figure object\Colormap, пометить один (1) для: 1 RED, 2 BLUE
- Стиль синуса/косинуса: Axes\ Compound\Polyline, выбрать Line solid 3 для обоих
- x/y оси: Axes\Text "x/y оси", режим фай ла включен, передний цвет 13, передний размер 3, срединное расположение осей, цвет сетки 13
- Название: текст "Синус и Косинус", размер шрифта 3, Цвет шрифта 13

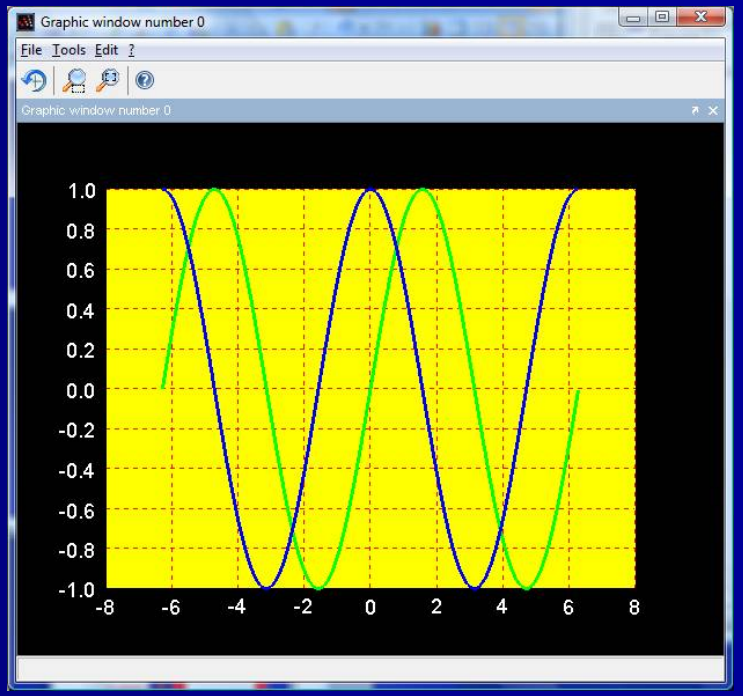


- **Стиль:** Передний размер **2 (осевой знак)**

Редактирование графического окна

- Рисунок редактор позволяет дать графическому редактору более красочный вид. Играя некоторое время со свойствами объектов редактора, и вы можете найти, например, следующие варианты:





Графическое командное окно

- Команда для создания нового графического окна для графика это:*

```
scf()
```

Для фигур:

```
show_window()
```

И устаревшие:**

```
xset()
```

- Окна, связанные с командами clear/ delete, например.:

```
clf()
xdel()
delete()
```

Устаревшие:

-
```
xclear()
xselect()
xbasc()
```
(Удаленные)

*) Единичные графики могут быть созданы без команды scf().

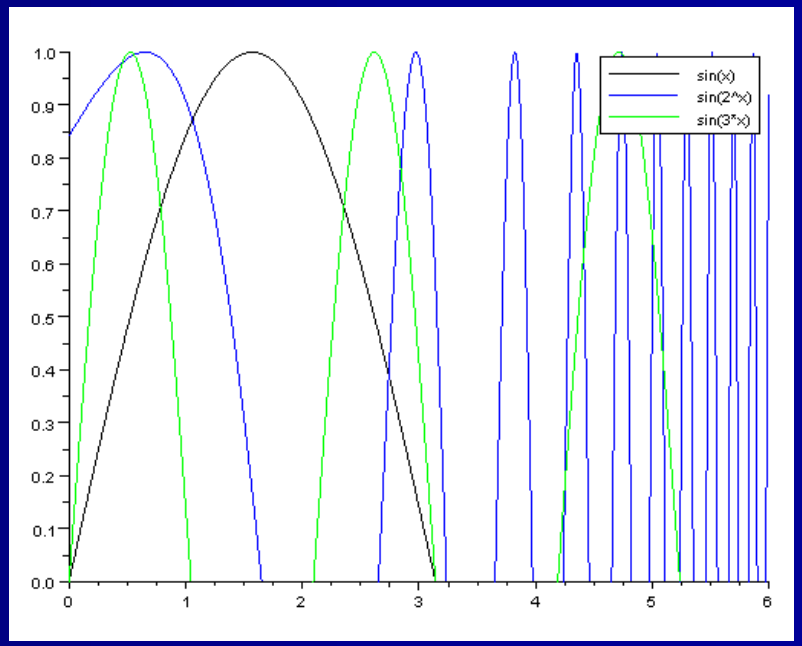
**) Устаревшие функции могут быть увидены в большинстве учебниках Scilab, но их следует избегать.

Команды Scilab начиная с x are связаны с графическим окном. История x возвращается к оконной системе X в Unix

Почему plot() и plot2d()?

- plot() и plot2d() создают 2D графики
- plot() заимствован из Matlab. Лица, имеющие опыт работы с Matlab могут использовать его (и, честно говоря, выгоды от plot2d() сомнительны)
- Scilab имеет дополнительную функцию plot2d(). Она предлагает больше вариантов для адаптации графика. Несколько графиков, например (напомним, однако, что несколько графиков были сделаны с помощью plot() в примере 1-2):

```
// multiple_plot.sce

// Demonstrates one alternative offered   /
// by the plot2d() function                /

clear,clc,clf;

x = [0:0.01:2*%pi]';
plot2d(x,[sin(x) sin(2^x)
sin(3*x)],rect=[0,0,6,1])
legend('sin(x)','sin(2^x)','sin(3*x)')
```



plot2d(): синтаксис

Примечание:
plot2d() **имеет также несколько иной старый синтаксис**

- Синтаксис plot2d() может быть использован как проводник для некоторых других графических команд, например для fplot2d() и histplot()
- plot2d() имеет следующие аргументы:

```
plot2d(logflag,x,y,optional arguments)
```

- x и y могут быть как векторы или матрицы, но с разными результатами для графика. logflag используется только с логарифмическими графиками, мы увидим это позже
- Множество дополнительных аргументов:

```
style, strf, leg, rect, nax
```

Осевые знаки и помеченные определения (векторы)

Стиль графы (цифровой)

Контроль отображения заголовков (по умолчанию "081")

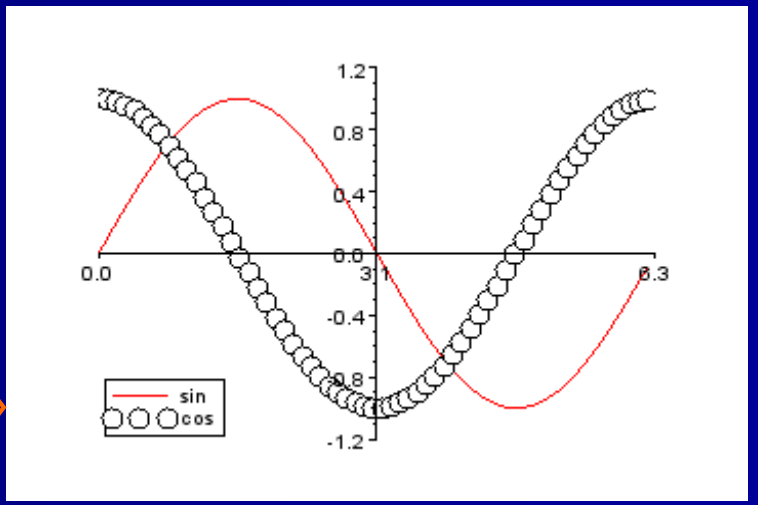
Легенда (строка, часто видели пустой (""))

Минимальные оценки для графика (векторные: [xmin, ymin, xmax, ymax])

plot2d(): демонстрация синтаксиса

```
// plot2d_demo.sce

clear,clc,clf;

x = 0:0.1:2*%pi;        // x axis definition
y1 = sin(x);            // Function 1
y2 = cos(x);            // Function 2

style1 = 5;             // "style" for sin
strf1 = '174';          // "strf"
rect = [0,-1.2,2*%pi,1.2];    // "rect"
nax = [4,%pi,4,7];            // "nax"
plot2d(x,y1,style1,strf1,' ',rect,nax)

style2 = -9;            // "style" for cos
strf2 = '000';          // No axes changes
leg = 'sin@cos';        // Legend definition
plot2d(x,y2,style2,strf2,leg)
```

- linspace() здесь не принимается
- style = 5 производит красный график
- leg пустой (' ') в графике синуса
- style=-9 производит круговые знаки
- Легенда добавляется в рисунке с помощью команды второго графика

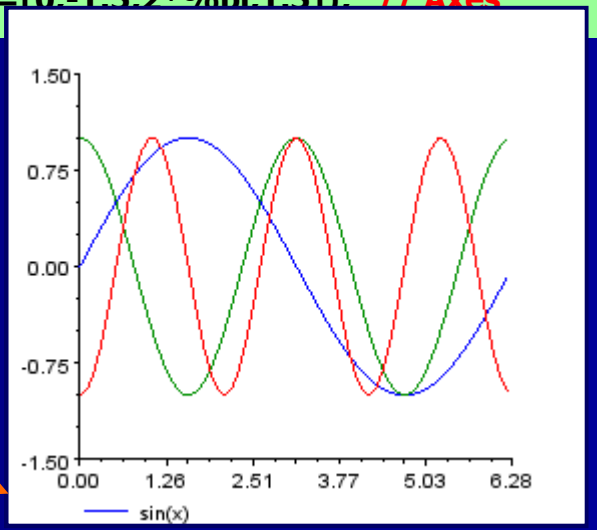


Scilab не может принять команду легенды как это было сделано здесь (ошибка?)

plot2d(): несколько графиков

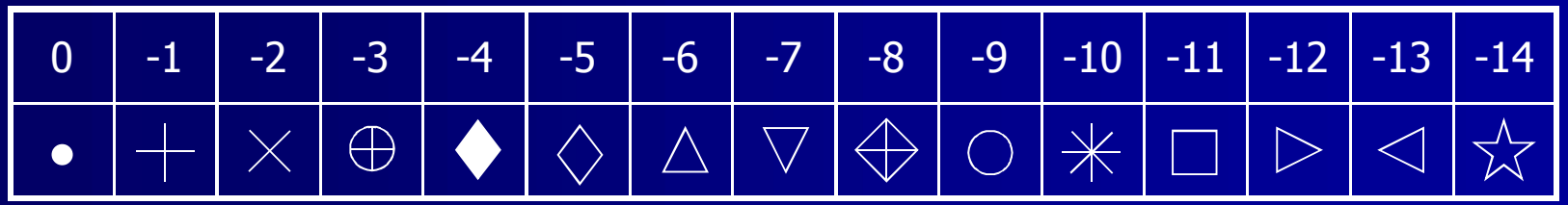
- Предыдущий слайд показали, как создать несколько графиков в одном окне с двумя отдельными командами plot2d()
- Несколько графиков могут быть объявлены в одном plot2d() с использованием векторного аргумента
- Дело, показанное здесь, также отличается от предыдущего наличием деклараций аргументов 'на месте'
- Scilab не правильно настроил график в окне; показана только легенда первого

```
// plot2d_multiple.sce

// Multiple graph declarations in a single  /
// plot2d() command                         /

clear,clc,clf();

x=[0:0.1:2*%pi]';
plot2d(x,[sin(x) cos(2*x) sin(3*x-%pi/2)],...
   [2,13,5],...                    // Graph colors
   leg="sin(x)@cos(x)@sin(3x)",... // Legend
   nax=[3,6,2,5],...               // Ticks & marks
   rect=[0,-1.5,2*%pi,1.5]);       // Axes
```



plot2d(): коды стиля

- Мы несколько раз сталкивались с цифровыми кодами для цветов графиков (стиль, число после x и y аргументов в `plot2d()`)
- Цветовыми обозначениями являются те, которые можно найти с помощью команды `getcolor()` на консоли. Наиболее важными из них являются 1 = черный, 2 = синий (9 = темно-синий), 3 = зеленый (13 = темно-зеленый), 5 = красный, 8 = белый, а 25 = коричневый
- На предыдущем слайде мы увидели, что код -9 создает круги. Подключите `getmark()` на консоли, чтобы увидеть список всех пользователей, в том числе коды для размеров марки, которые можно использовать с ручными командами. Есть во всех 15 этих марках (всегда черные):

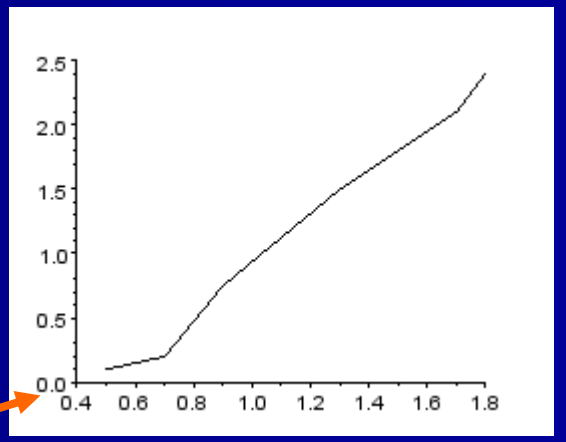


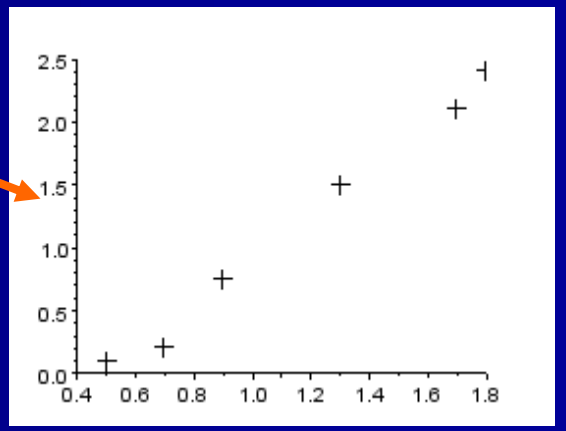
plot2d(): демонстрация с матрицами

Простой скрипт ниже демонстрирует команду plot2d(), когда аргументы x и y матрицы, и стиль 1 и -1

scf() используется, чтобы открыть новое графическое окно. В противном случае + знаки второй команды plot2d() дожны быть в верхней части первой

```
-->x = [.5 .7 .9 1.3 1.7 1.8]';

-->y = [.1 .2 .75 1.5 2.1 2.4]';

-->plot2d(x,y, style=1)

 -->scf();

-->plot2d(x,y, style=-1)
```





Команда понятнее если аргументы написаны на простом (стиль=-1) но, как показанно в прошлых демонстрациях, одних чисел достаточно

fplot2d()

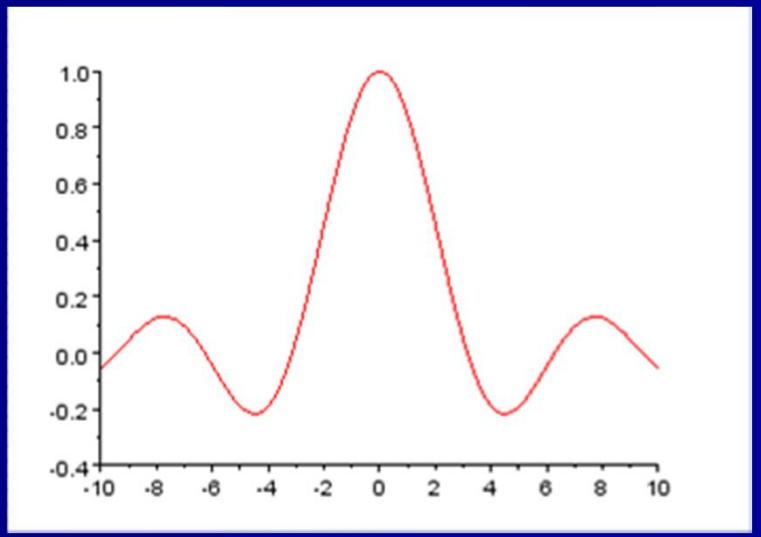
- fplot2d() это вариация plot2d()
- С fplot2d() функция и ее определения могут быть включены в аругменты
- Общая форма fplot2d() является:

  fplot2d(x,f,opt arguments)

- Демо справа показывает случай когда
  - x = linspace(-10,10,100)
  - f = функция sinc встроена в Scilab
  - Стиль=5 является необязательным аргументом
- Существует также 3D альтернатива fplot3d()

```
-->fplot2d(linspace(-10,10,100),sinc,style=5)
```

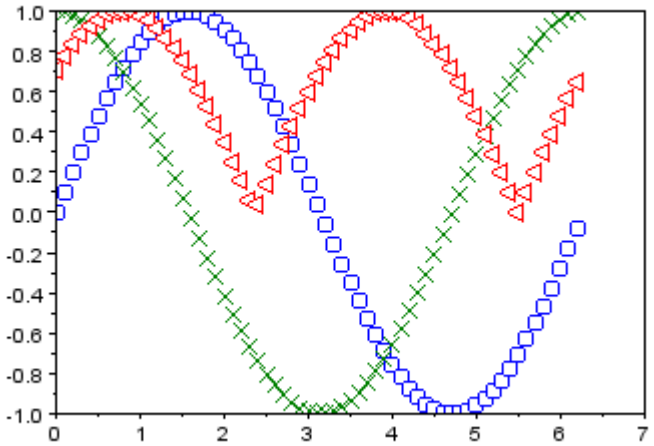
x f Необ аргум



plot(): красота прстоты (1/2)

- Matlab/Scilab's функция plot()* предлразличать несколько графиков чем это делает plot2d(). Это с помощью клавиатуры символов так, как это делалось на телетайпах пол века назад
- Вот три графика построены с помощью одной команды plot(). Определения стиля 'o', 'x' и '<.' Обратите внимание, что t повторяется для каждого графика
- Видно, что '<' (красный) дает треугольник, который указывает в направлении пути линии

```
// plot()_demo.sce
// Demonstration of plot() syntax /

clf();

t=0:0.1:2*%pi;
plot(t,sin(t),'o',..          // Plot with 'o'
   t,cos(t),'x',..            // Plot with 'x'
   t,abs(sin(t+%pi/4)),'<')   // Plot with '<'
```



***) Функция plot() в Scilab не поддерживает все свойства properties как его коллега в Matlab**

plot(): красота простоты (2/2)

Следующий список содержит основные коды стилей линий для plot():

-	Сплошная линия (по умолчанию)	x	Крест
--	Пунктирная линия	'Площадь' or 's'	Площадь
:	Линия отрыва	'Алмаз' or 'd'	Алмаз
-.	Пунктир с точками	^	Выглядит как треугольник
+	Знак плюс	v	Направленный вниз треугольник
○	Круг	>	Направленный вправо треугольник
*	Звездочка	<	Направленный влево треугольник
.	Точка	'Пентаграмма'	Пятиконечная звезда

Аргументы цвета: k – черный, w – белый, r - красный, g - зеленый, b – синий, c – голубой, m – пурпурный, y – желтый (символ должен быть перед кодом стиля, внутри одинарных или двойных кавычек, например, 'r+')

3D графики: plot3d()

- Синтаксис в plot3d() очень похож на тот, что и в plot2d(). В дополнение к обязательным аргументам x,y,z, функция plot 3d() может среди других возможностей иметь следующие аргументы:

  plot3d(x,y,z,theta,alpha,leg,flag,ebox)

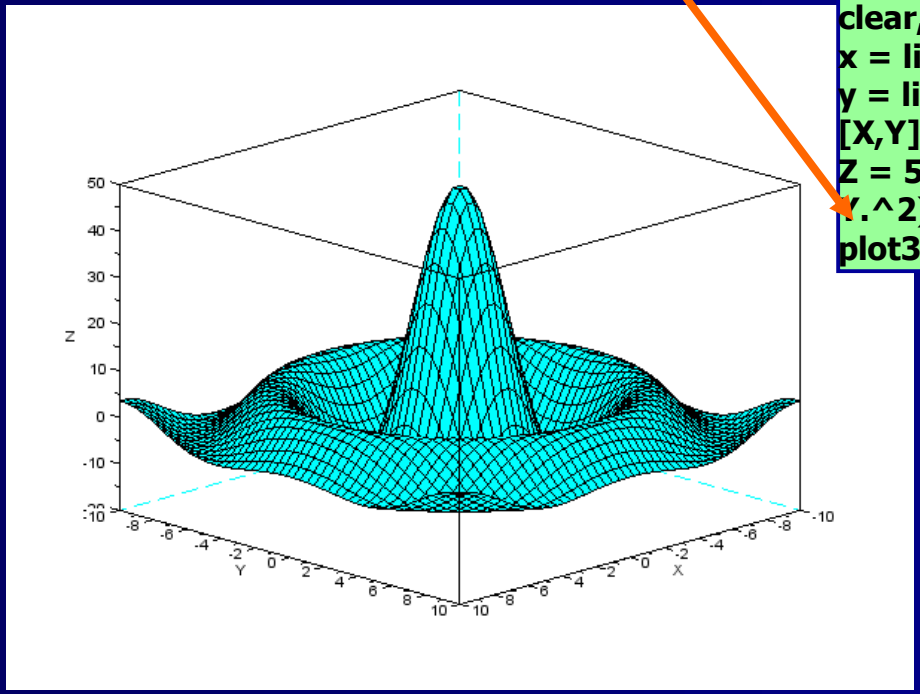
- Проверте объяснения в справке
- Нже мы построим 3D график функции sin(x)/x, использую некоторые из поверхностных возможностей определения plot3d()
- Scilab определяет только 2D функцию sinc(x), чтобы перейти к 3D мы будем применять выражение

  $r = \sqrt{(x^2 - y^2)}$

- Из упомянутых выше аргументов мы будем использовать leg="X@Y@Z" знаки x,y, и z оси и flag=[mode,type,box] чтобы определить цвет поверхности, масштабирование и рамка графика

3D графики: plot3d(), Скрипт и график для 3D sinc()

Обратите внимание на [X,Y] = ndgrid(x,y) и использование оператора точчки в Z




```
// sinc3D.sce
// Plot the sinc function (sin(x)/x) using plot3d()
/
// with surface definition arguments                /

clear,clc,clf;
x = linspace(-10,10,50);
y = linspace(-10,10,50);
[X,Y] = ndgrid(x,y);        //Create array for xy grid
Z = 50*sin(sqrt(X.^2 + Y.^2))./sqrt(X.^2 +
Y.^2);
plot3d(x,y,Z,leg="X@Y@Z",flag=[4,2,4])
```

Измените plot3d() на plot3d1() чтобы получить различные текстуры

Другой подход к решению этой задачи показан в Example 3-5. Существует ошибка в скрипте, приведенной в Help/meshgrid

3D графики: surf(), Задача и сценарий

- Напишите сценарий что функция графика
  $z = (2*x^2 - y^2)\exp(-x^2 - 0.5*y^2)$,
  Где $-2 \le x \le 2$ и $-3 \le y \le 3$

- Функция $\mathrm{linspace(a,b,m)}$ создает линейно расположенные x и y строк векторов (“от a до b с шагом m”)
- Используя векторы x и y, команда $[X,Y] = \mathrm{meshgrid(x,y)}$ создает 2D матрицу в xy-плоскости
- Создать Z-значения для каждого элемента 2D матрицы
- График в результате 3D функции

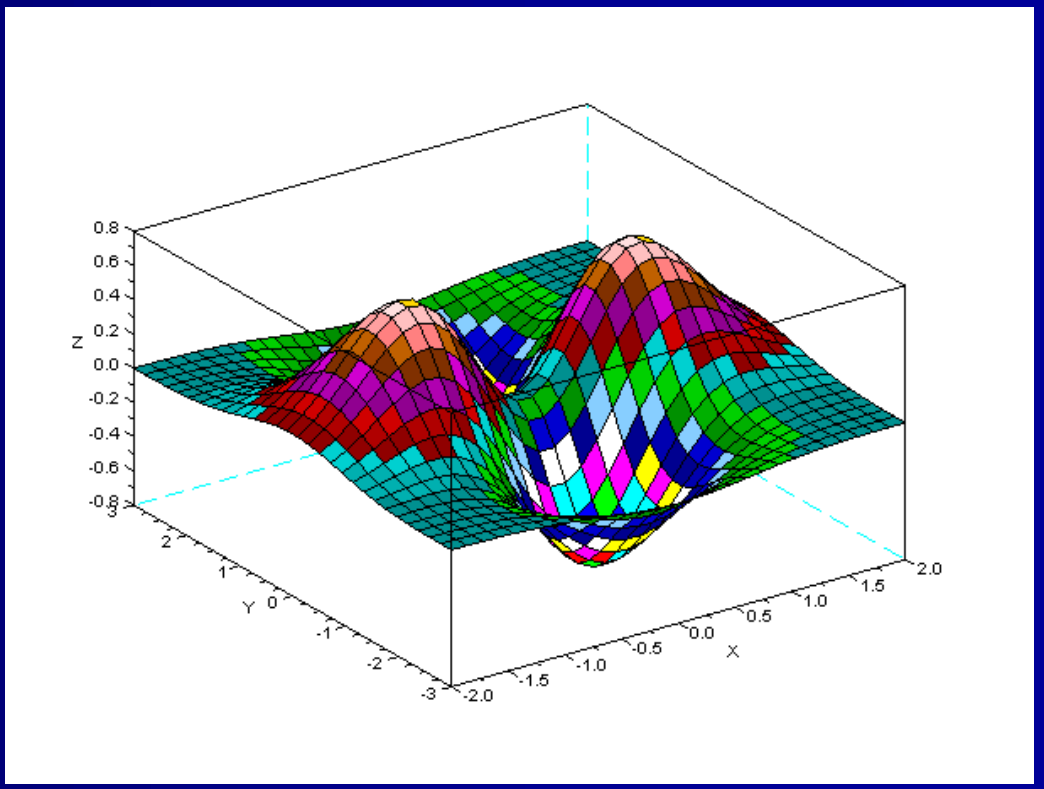
```
// surf_ex1.sce

// Plot the function                              /
// z=(2x^2 - y^2)exp(-x^2 - 0.5y^2)
/
// for -2<x<2, -3<y<3, where <
indicates  /
// "less than or equal to"                        /

clear,clc,clf;
x=linspace(-2,2,30);    // Linear spacing
y=linspace(-3,3,30);
[X,Y]=meshgrid(x,y);   // Surface mesh
Z=(2*X.^2-Y.^2).*exp(-X.^2-0.5*Y.^2);
surf(X,Y,Z)                  // Plot 3D surface
```

3D графики: surf(), график

Разве это не мило!

Цвета могут быть не так уж велики, но они могут быть изменены с помощью ручных команд. Это будет показано в Example 3-5

surf() имеет параллельную форму под названием mesh() который используется таким же образом, как surf(), но в нем отсутствует затенение

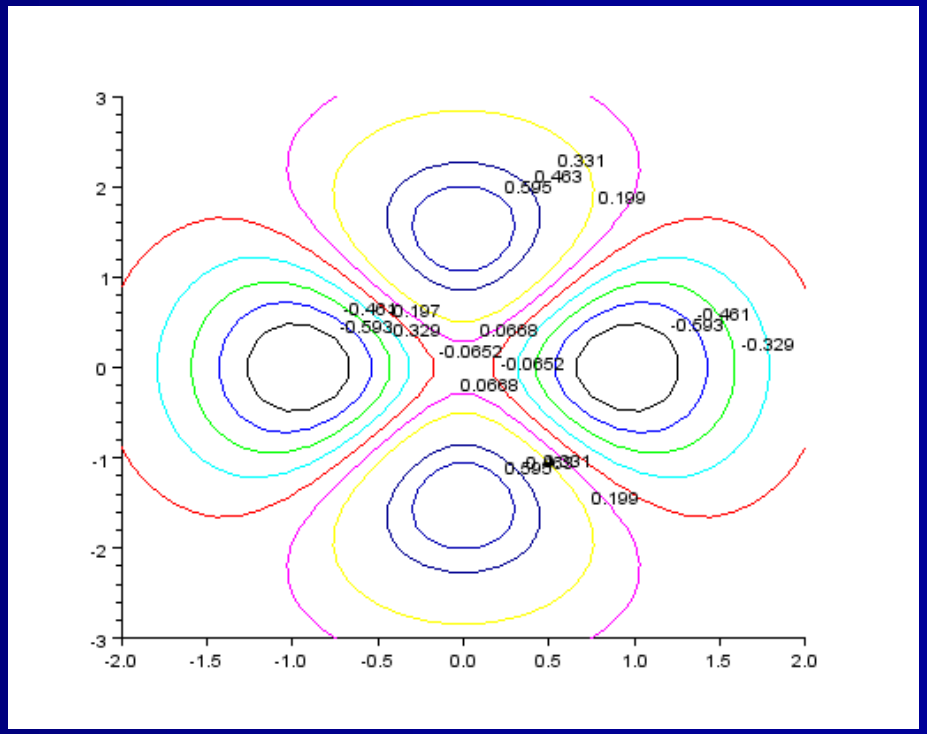


Если вы нажмете на кнопку отображения для surf() в Help Browser, Scilab сначала отображает ряд альтернатив и затем все разбивает.

Контурные графики: contour()

- Вернемся к выражению $z = (2*x^2 - y^2)\exp(-x^2 - 0.5*y^2)$, и график его 2D контура (уровень/высота кривых)
- Для этого только требуется изменить команду скриптов графика

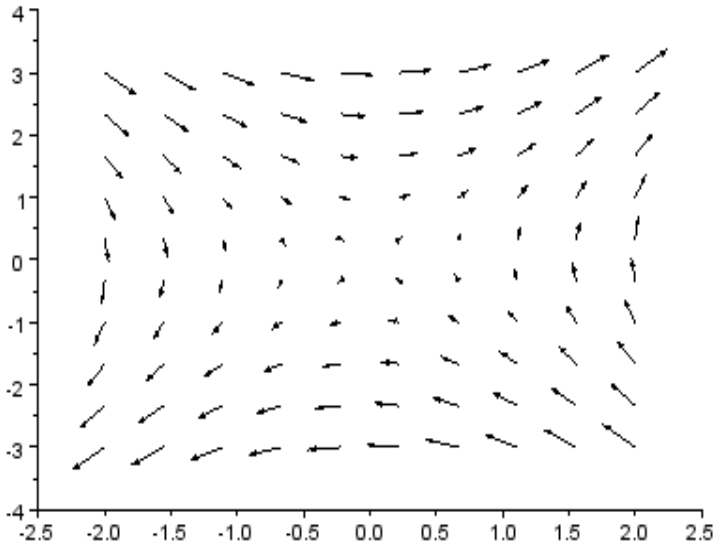
```
// contour.sce

// Plot the 2D height curves for the          /
// function                                   /
// z=(2x^2 - y^2)exp(-x^2 - 0.5y^2)           /
// for -2<x<2, -3<y<3, where < indicates      /
// "less than or equal to"                    /

clear,clc,clf;
x=linspace(-2,2,30);
y=linspace(-3,3,30);
[X,Y]=meshgrid(x,y);
Z=(2*X.^2-Y.^2).*exp(-X.^2-0.5*Y.^2);
contour(x,y,Z,10)
```



Векторные поля: champ()

- Векторное поле 2D для выражения $z = (2*x^2 - y^2)\exp(-x^2 - 0.5*y^2)$ могут быть визуализированы, изменив выражение графика на champ(), и регулирования промежутков функцией linspace():

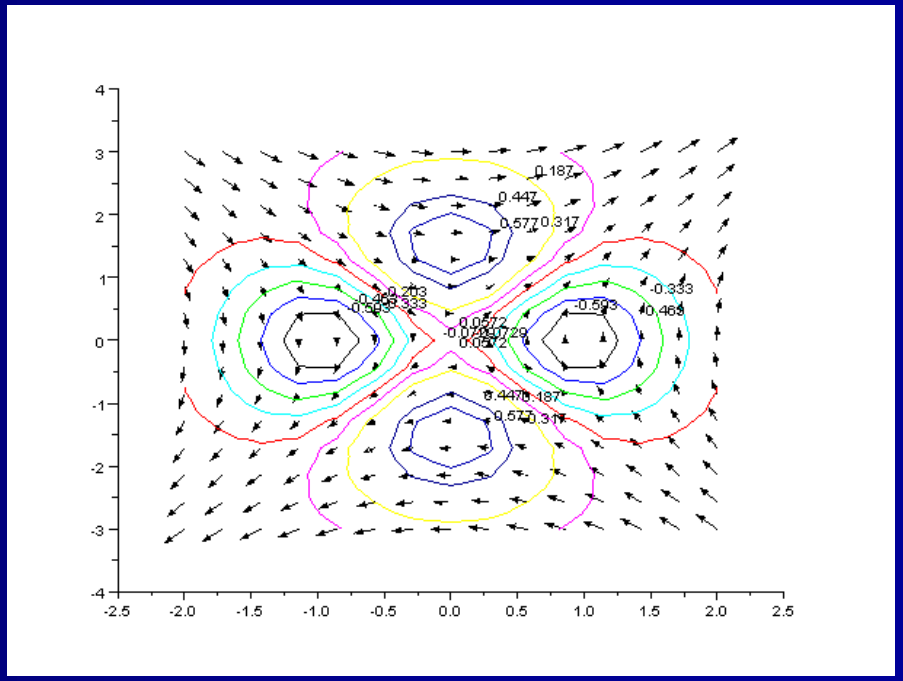
```
// vector_field.sce

// Plot the 2D vector fields for the function /
// z=(2x^2 - y^2)exp(-x^2 - 0.5y^2)       /
// for -2<x<2, -3<y<3, where < indicates  /
// "less than or equal to"                /

clear,clc,clf;
x=linspace(-2,2,10);
y=linspace(-3,3,10);
[X,Y]=meshgrid(x,y);
Z=(2*X.^2-Y.^2).*exp(-X.^2-0.5*Y.^2);
champ(x,y,X,Y)
```



Смешанные контуры и векторные поля

- Векторные поля не очень информативны по себе, но ситуация улучшается, когда они слиты с контурами
- В предыдущем случае, просто вставьте команды champ() и contour() в одном сценарии и вы получите их в одном графике:

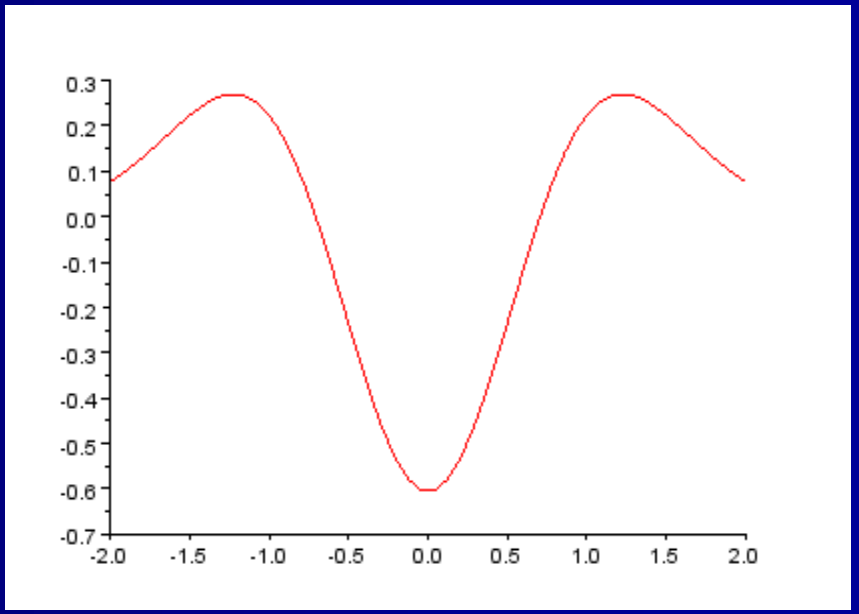
```
// contour-vector.sce
// Plot the combined contour and vector
/
// fields for the function                    /
// z=(2x^2 - y^2)exp(-x^2 - 0.5y^2),
/
// for -2<=x<=2, -3<=y<=3
/clf;
x=linspace(-2,2,15);
y=linspace(-3,3,15);
[X,Y]=meshgrid(x,y);
Z=(2*X.^2-Y.^2).*exp(-X.^2-0.5*Y.^2);
champ(x,y,X,Y)
contour(x,y,Z,10)
```



Резка 3D поверхности

- Мы видим контуры 3D поверхности $z = (2*x^2 - y^2)\exp(-x^2 - 0.5*y^2)$ в определенной плоскости, определяя план в случае (ниже y = -1) и, вернувшись к 2D черчению:

```
// cutting.sce

// Cut the the function                    /
// z=(2*x^2 - y^2)exp(-x^2 - 0.5*y^2)
/
// along the plane y = -1                  /

clf;
x=linspace(-2,2,50);
y=linspace(-1,-1,0);
[X,Y]=meshgrid(x,y);
Z=(2*X.^2-Y.^2).*exp(-X.^2-0.5*Y.^2);
plot2d(X,Z,5)
```



Смешанные 2D/3D графики (1/2): сценарий

Scilab имеет свои собственные идеи о том, что он должен делать, если команда contour() добавляется к сценарию 3D-команды графика (plot3d(), surf()). Методом проб и ошибок требуется

Вопрос: следует ли contour() до или после plot3d()?

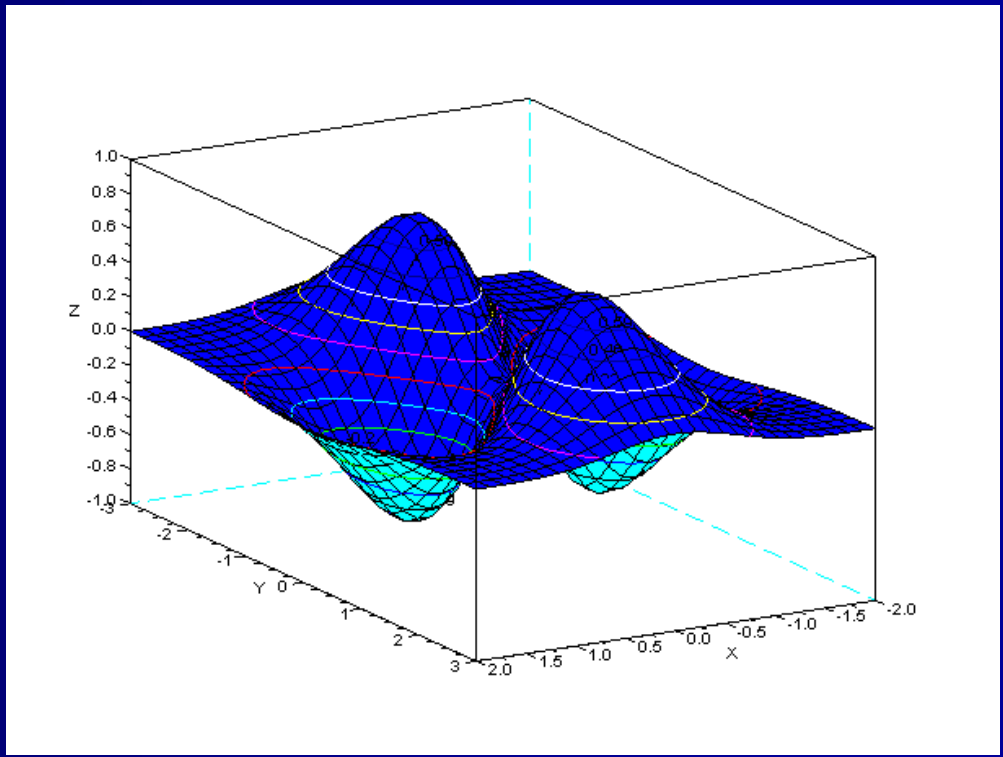
Ответ: Scilab принимает обе альтернативы, но с резко различными результатами

Только первый flag[] аргумент contour() имеет влияние на график

```
// plot3d-contour.sce

// Plot the combined 3D graph and contour  /
// of the function                         /
// z=(2x^2 - y^2)exp(-x^2 - 0.5y^2),       /
// for -2<=x<=2 and -3<=y<=3               /

clear,clc,clf;
x=linspace(-2,2,30);
y=linspace(-3,3,30);
[X,Y]=meshgrid(x,y);
Z=(2*X.^2-Y.^2).*exp(-X.^2-0.5*Y.^2);   // Same as
before
contour(x,y,Z,10,flag=[0,0,0]);       // First flag[]
argument
plot3d(x,y,Z,theta=60,alpha=80);    // Turn 60 and 80 deg
```

Смешанные 2D/3D графики (2/2): график

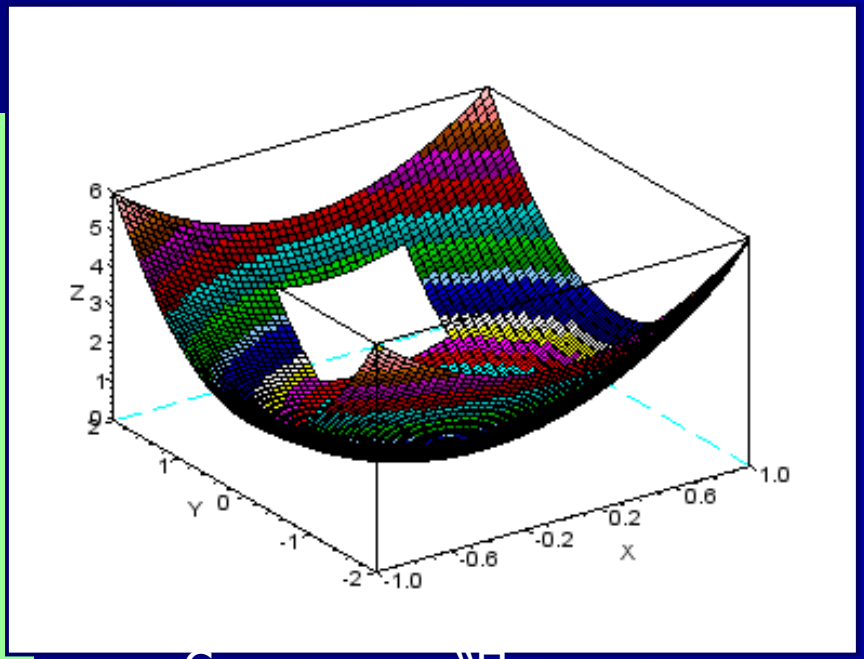
Поверхность выглядит иначе, когда она была построена с использованием `surf()`. Причина в том, что оси x и y инвертируются по сравнению с предыдущим случаем
Нет смысла отрицать, что не остается нерешенных вопросов о поведении Scilab в этом случае



3D график с отверстием

Функция %nan позволяет определить величины z, которые должны быть исключены из 3D графика:

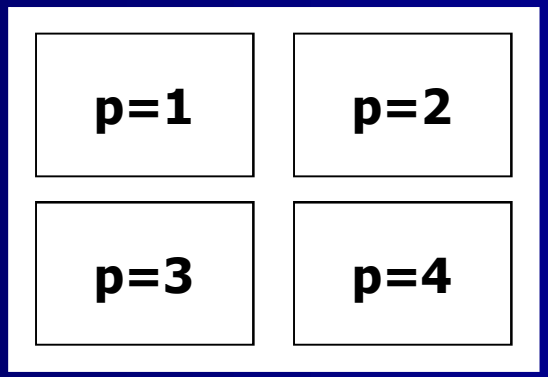
```
// hole.sce

// 3D surface with a hole punched   /
// into it with the %nan command    /
// (z values not to be represented)  /

clear,clc,clf;

function z = f(x, y)
   z=2*x^2+y^2;
endfunction

x = linspace(-1,1,50);
y = linspace(-2,2,100);
z = (feval(x,y,f))';          // Evaluate function
z(75:90,20:35) = %nan;  // Definition of hole
surf(x,y,z)                    // Plot surface
```



Существует "Польская логика" за аргументами z которая спрашивается для испытания и ошибки чтобы получилось верно

subplot()

- Subplots является одним из способов представления нескольких графиков на одном кадре
- Функция $\mathrm{subplot(m,n,p)}$, или $\mathrm{(mnp)}$, разбивают графическое окно на m строк и n столбцов, и subplot в случае занимает позиции p. В случае четырех подокон, $\mathrm{subplot(22p)}$, положение p как показано:



- Мы сделаем это для $z = (2*x^2 - y^2)\exp(-x^2 - 0.5*y^2)$, путем слияния ранее четырех случаев в одном кадре

subplot(): демонстрация сценария

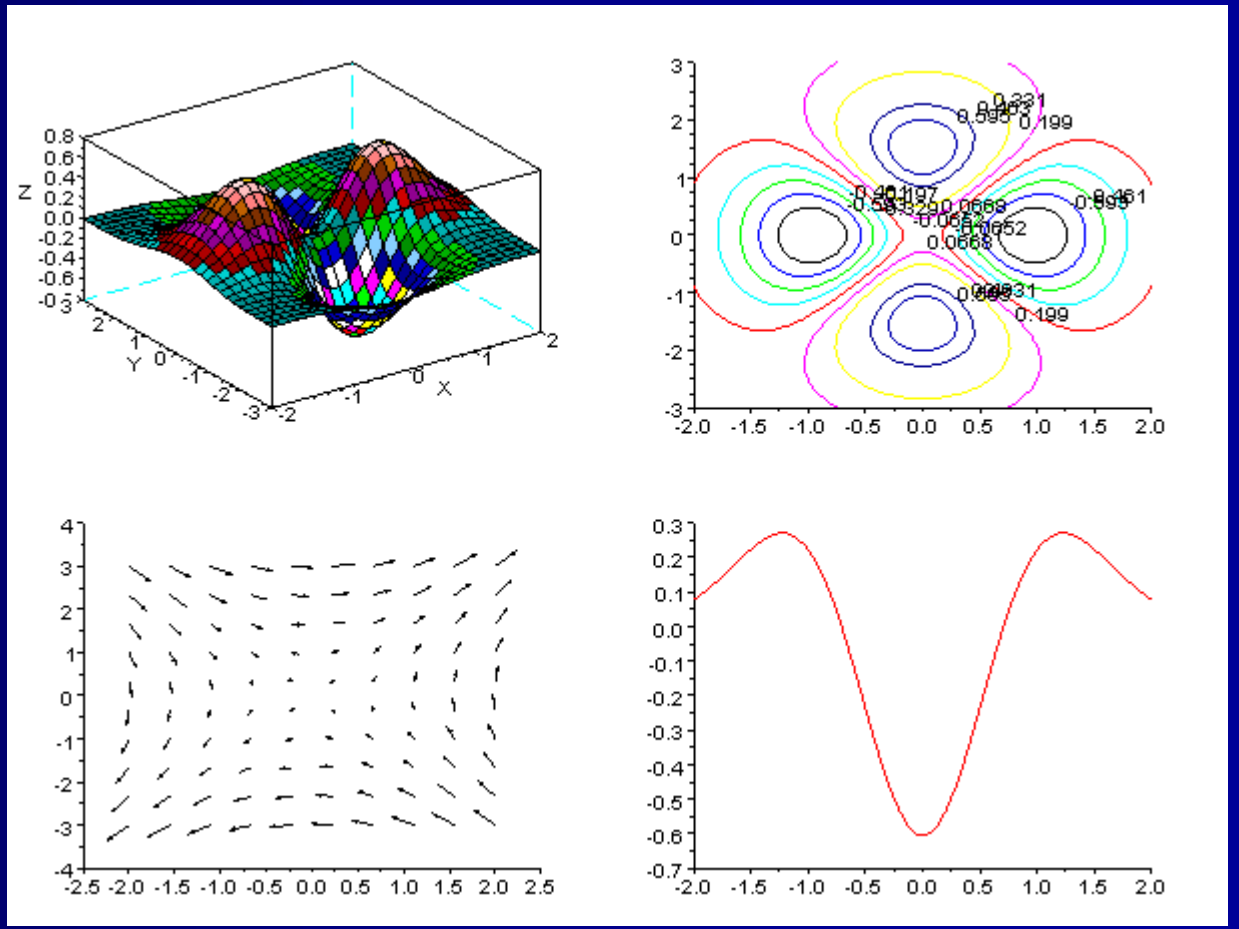
```
// subplot.sce

// Presents different aspects of                /
// the function                                 /
// z=(2x^2 - y^2)exp(-x^2 - 0.5y^2)     /
// in four subplots                             /

clear,clc,clf;
x=linspace(-2,2,30);
y=linspace(-3,3,30);
[X,Y]=meshgrid(x,y);
Z=(2*X.^2-Y.^2).*exp(-X.^2-0.5*Y.^2);
subplot(221)
surf(X,Y,Z)

subplot(222)
contour(x,y,Z,10)
```

```
x=linspace(-2,2,10);
y=linspace(-3,3,10);
[X,Y]=meshgrid(x,y);
Z=(2*X.^2-Y.^2).*exp(-X.^2-0.5*Y.^2);
subplot(223)
champ(x,y,X,Y)

x=linspace(-2,2,50);
y=linspace(-1,1,0);
[X,Y]=meshgrid(x,y);
Z=(2*X.^2-Y.^2).*exp(-X.^2-0.5*Y.^2);
subplot(224)
plot2d(X,Z,5)
```

Следует отметить, что только график функции был повторен для (222)

subplot(): демонстрация графика

Существует еще одна функция для сюжетных линий: xsetech(). **Проверьте** Help **для получения деталей**



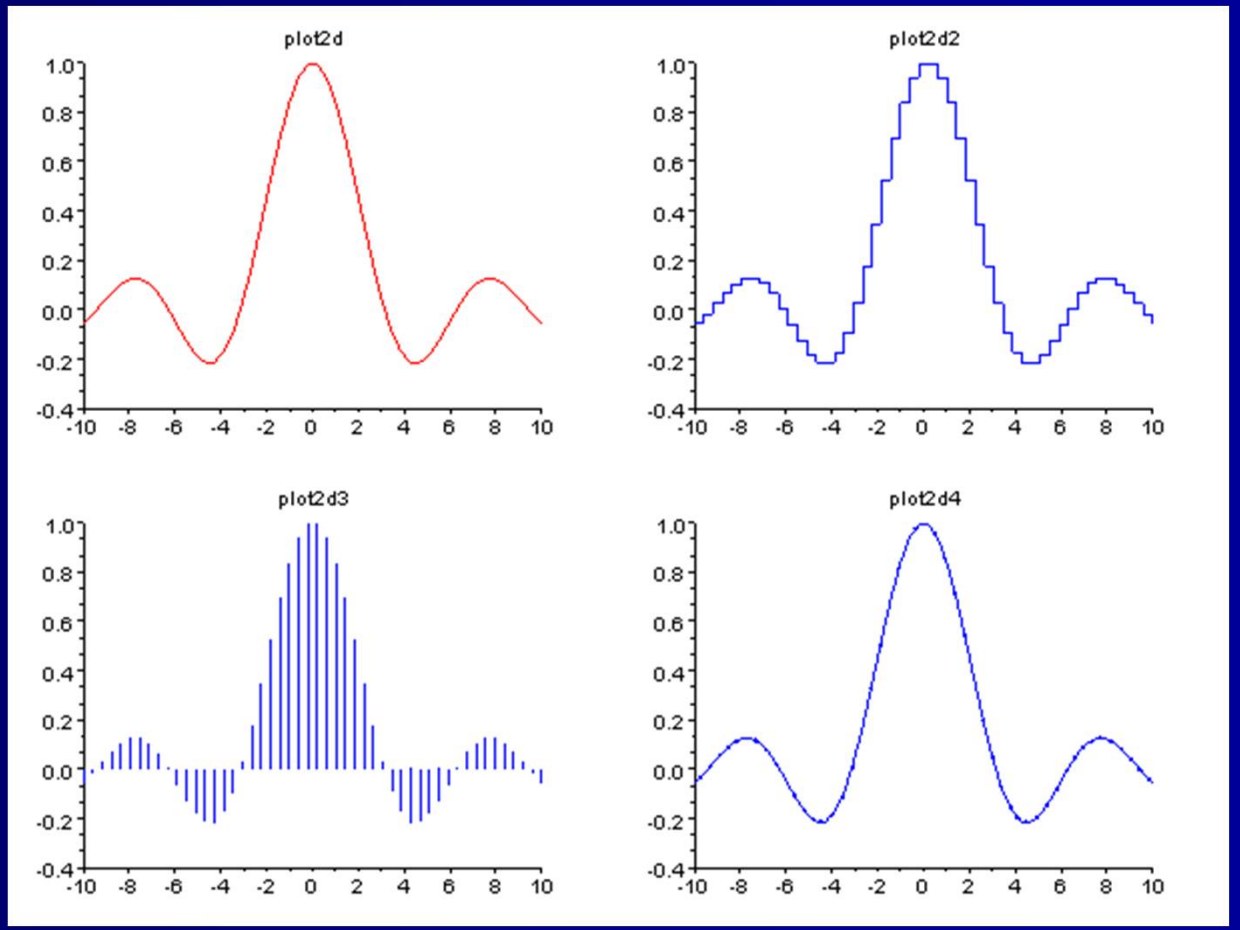
plot2d2(), plot2d3(), plot2d4(): демо, сценарий

- Функция plot2d() имеет три варианта:
- plot2d2() для ступенчатых функций
- plot2d3() для вертикальных полос
- plot2d4() для линии в виде стрелы
- Эффект этих команд на функцию sinc() показан на следующем слайде

```
// plot2dx.sce

// Demonstration of the basic sinc function plotted    /
// with plot2d(), plot2d2(), plot2d3(), and plot2d4()   /

clear,clc,clf;
x = linspace(-10,10,50);

subplot(221);
plot2d(x,sinc(x),style=5)   // Plot continuous line
xtitle('plot2d')

subplot(222);
plot2d2(x,sinc(x),style=2)  // Plot with steps
xtitle('plot2d2')

subplot(223);
plot2d3(x,sinc(x),style=2)  // Plot vertical bars
xtitle('plot2d3')

subplot(224);
plot2d4(x,sinc(x),style=2)  // Plot arrow style
xtitle('plot2d4')
```

plot2d2(), plot2d3(), plot2d4(): демо, график

Примечание: Вы все еще можете увидеть устаревшее plot2d1() в руководствах. Вместо этого следует использовать plot2d() (хотя, plot3d1() устаревшей не была объявлена)



Гистограммы: функции для их создания

- Гистограммы графически обычно представляются прямоугольниками одномерных данных
- Основная функция Scilab для построения гистограмм:
  `histplot(x,data,opt_arguments)`
- Диаграммы полос, распространенной формой гистогамм, дается:
  `bar(x,y,width,color,style)`
  Или, для горизонтальных полос:
  `barh(x,y,width,color,style)`
- Трехмерные диаграммы полос могут быть созданы с помощью команды:
  `hist3d(z,opt_arguments)`
  И с добавлением `x` и `y` векторов:
  `hist3d(list(z,x,y),opt_arguments)`
- Используйте `Help` чтобы узнать детали

Гистограммы: демо, сценарий

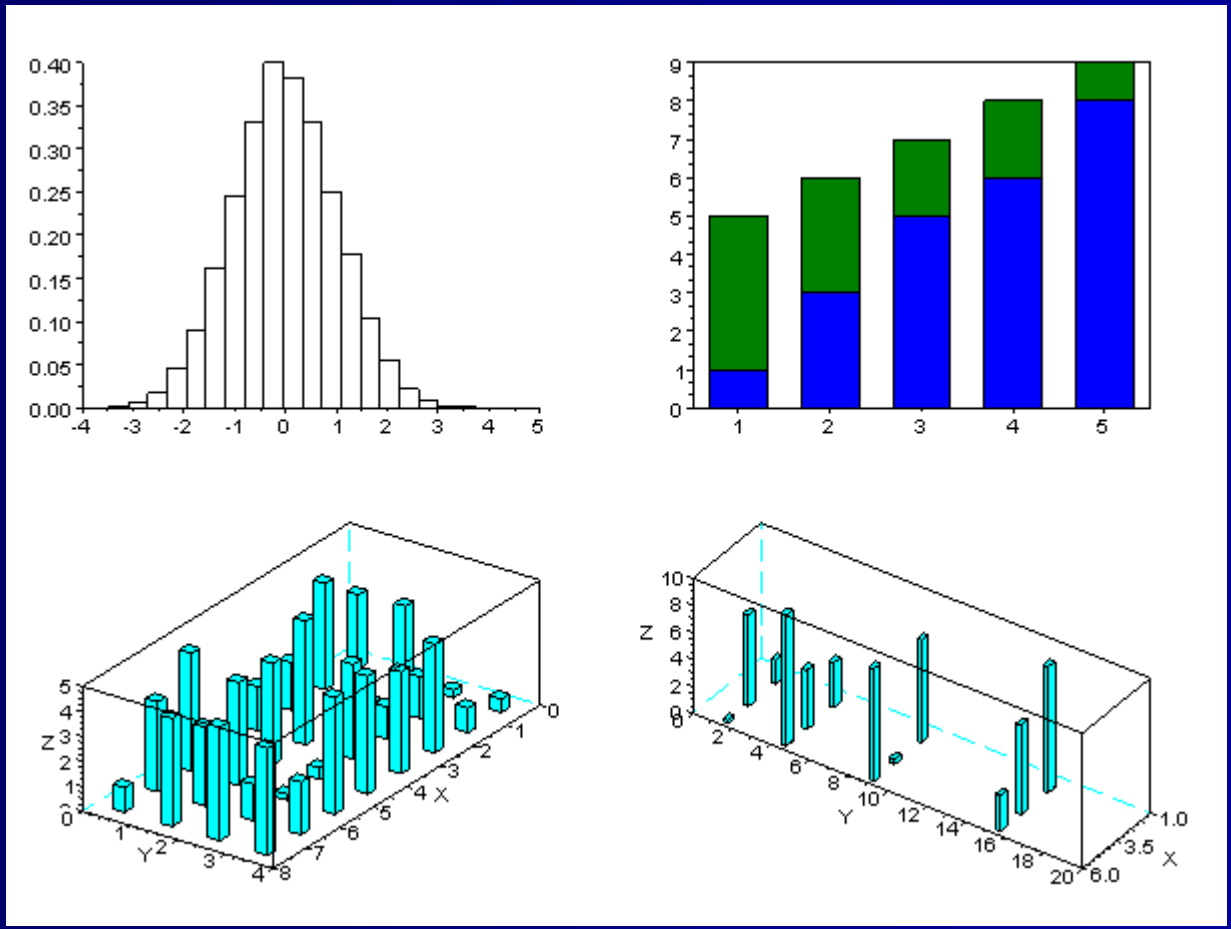
- Сценарии ниже, предназначены для демонстрации различных типов гистограмм, представленных как (22P) подграфики

```
// histogram_subplot.sce

// Demonstration of histogram types   /
// using subplots                     /

clear,clc,clf;
subplot(221)
data=rand(1,10000,'normal');
histplot(20,data)      // Traditional histogram

subplot(222)
y=[1 3 5 6 8];
z=[y;4 3 2 2 1]';      // Transpose necessary!
bar(z,0.7,'stacked')  // "on top of each other"
```

```
subplot(223)
hist3d(5*rand(8,4))  // 3D histogram

subplot(224)
z=10*rand(3,4);
x=[1 3 5 6];
y=[1 2 7 11 20];
hist3d(list(z,x,y))     // 3D hist, add x/y vectors
```

Аргумент list() определяет распределение случайных величин z над плоскостями x и y

Гистограммы: демо, графики



Старый синтаксис графиков (1/2): демо, сценарий

Графика синтаксис Scilab изменилась с версии 3.1. Это демо показывает старый синтаксис `plot2d()` в случае с тремя графиками,

`y1=f(x1)`, `y2=f(x2)` и `y3=f(x3)`,

и в том же кадре

Обратите внимание на определение кадра и сравните с методом, использованным в Example 1-2

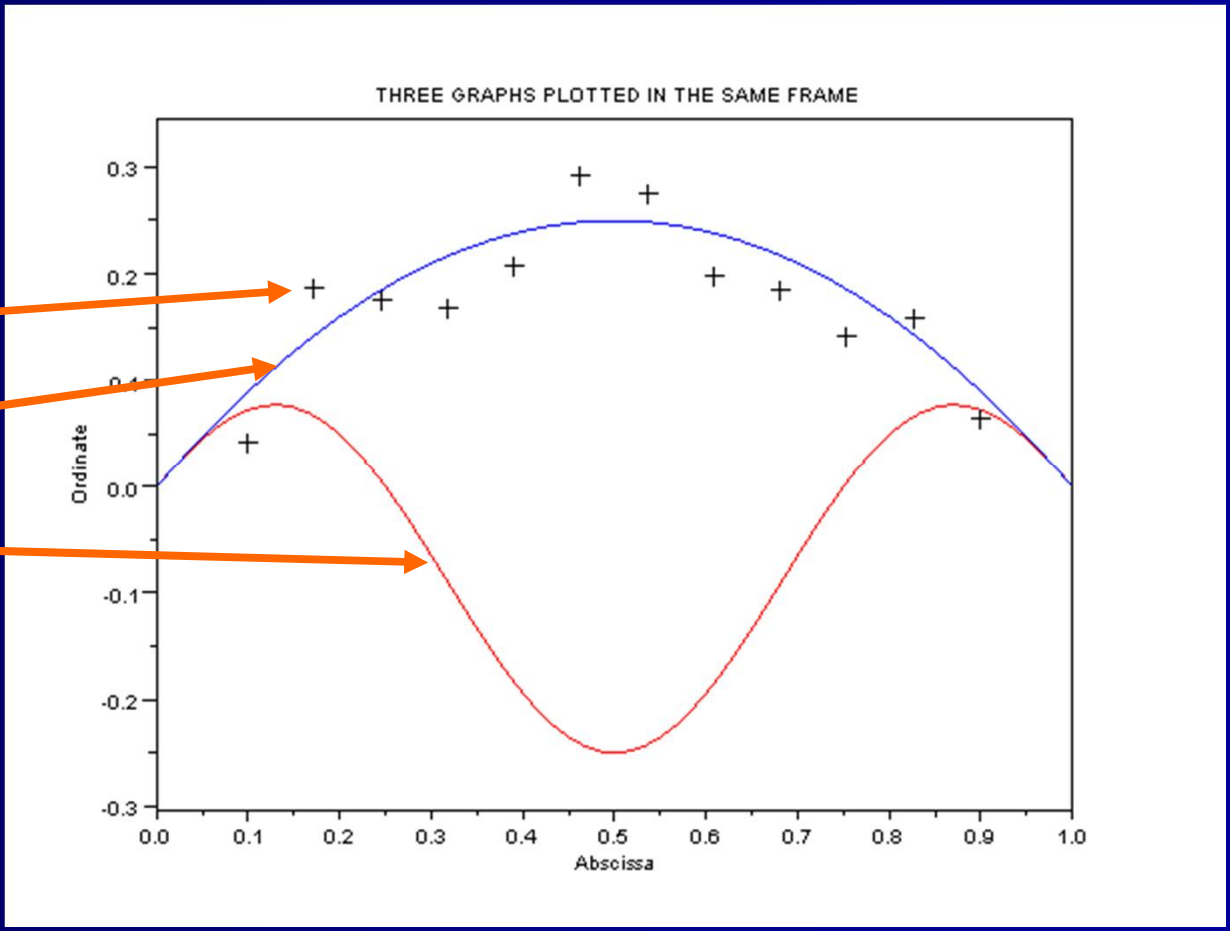
```
// multiple_plots2.sce
// Demonstration of a method for producing  /
// three plots y1=f(x1), y2=f(x2),y3=f(x3)   /
// in the same frame. Note how the frame    /
// is defined                                /

clear,clc,clf;
x1 = linspace(0,1,61);
x2 = linspace(0,1,31);
x3 = linspace(0.1,0.9,12);
y1 = x1.*(1-x1).*cos(2*%pi*x1);        // First graph
y2 = x2.*(1-x2);                       // Second graph
y3 = x3.*(1-x3) + 0.1*(rand(x3)-0.5);  // Third, as y2 with
disturbance

ymin = min([y1,y2,y3]);      // Select minimum to define frame bottom
ymax = max([y1,y2,y3]);      // Select maximum to define frame top
dy = (ymax - ymin)*0.1;      // Border for min/max
rect = [0,ymin - dy,1,ymax+dy];        // Frame limits, start at 0
plot2d(x1,y1,5,"011"," ",rect)         // First call with frame
definitions
plot2d(x2,y2,2,"000")        // Second call, only type/color (2)
definition
plot2d(x3,y3,-1,"000")       //  Third call, defines marks(-1)
xtitle("THREE GRAPHS PLOTTED IN THE SAME
FRAME","Abscissa","Ordinate")
```

Старый синтаксис графиков (2/2): демо, график

y3

y2

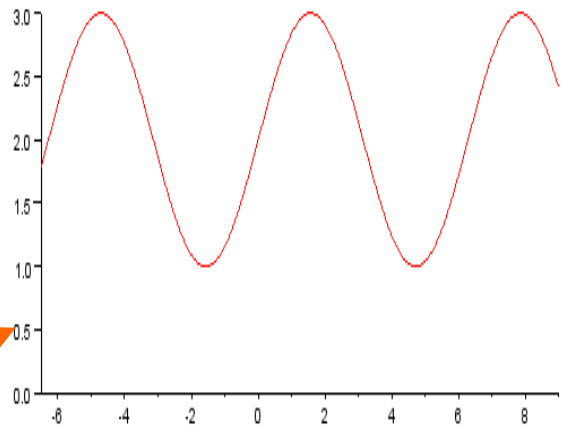
y1

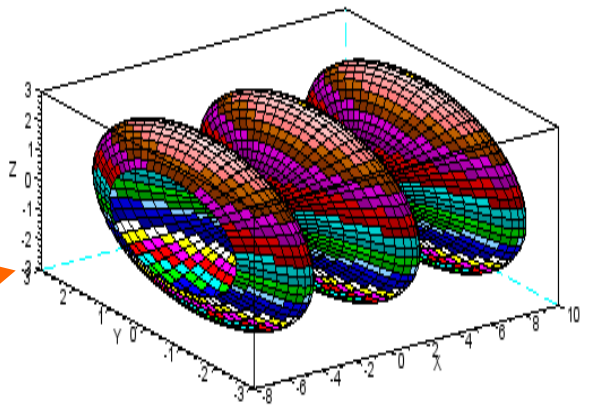


Поверхности вращения

Поверхность вращения создается путем умножения первоначальной функции, которая переопределяется как 2 +sin(T), по .*sin(PHI) и .*cos(PHI)

```
// rotation_surface.sce

// Plot the rotation surface created by   /
// the function y=2+sin(x) as it rotates  /
// around the x-axis                      /

clear,clc,clf;

// Define function to rotate:
//-----------------------------------
x=-10:.01:10;
subplot(211)
plot2d(x,2+sin(x),5,rect=[-6.5,0,9,3])

// Rotate 2+sin(x) around y-axis:
//-------------------------------------------
t=linspace(-6.5,9,60);
phi=linspace(0,2*%pi,60);
[T,PHI]=meshgrid(t,phi); // Create mesh
X=T;
Y=(2+sin(T)).*sin(PHI);
Z=(2+sin(T)).*cos(PHI);
subplot(212)
surf(X,Y,Z)
```





Логарифмическая шкала: задача и сценарий

- Постройте диаграмму Боде для функции

$$G(s) = \frac{100}{(s-10)(s-90)}$$

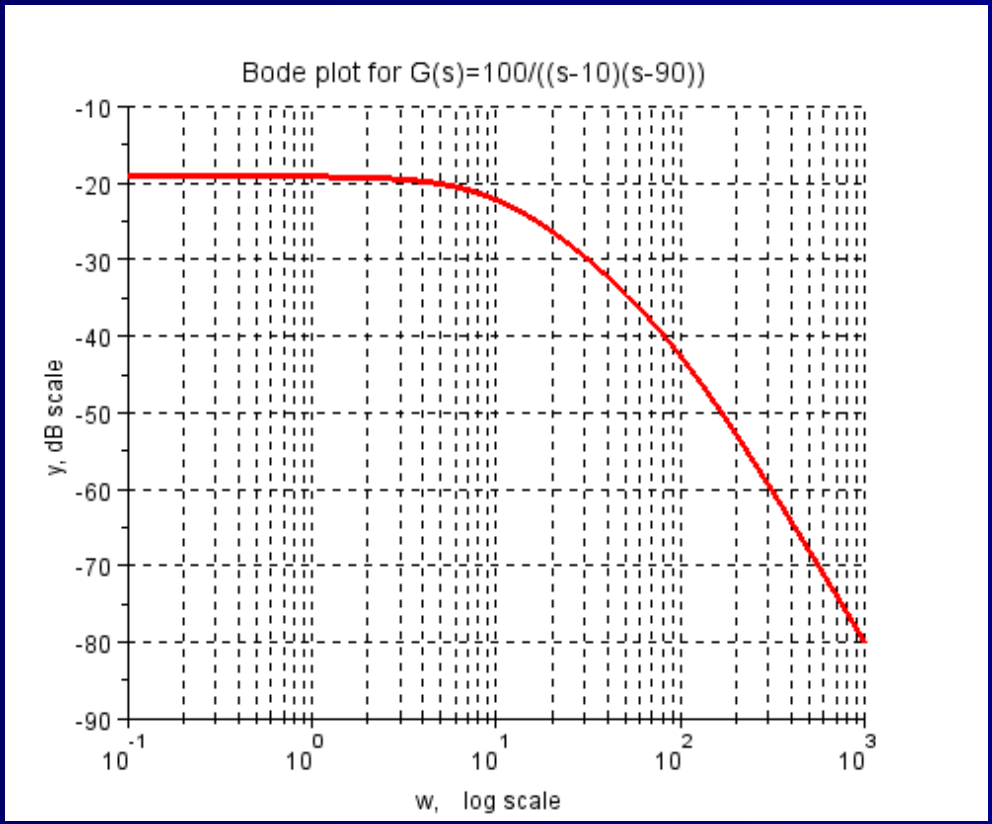
Где s = iω и угловая частота ω = 0.1 … 1000

- Примечательные двойные точки `100../(s-10)` в команде G. Первая точка является десятичной, затем появляется оператор точки
- Поставить логарифмическую ось ω горизонтально, а децибел масштаба y=20(|G(ωi)|) вертикально

```
// log_plot.sce

// Plot the Bode diagram for the function       /
// G(s) = 100/((s-10)(s-90)). Use the normal  /
// logarithmic x-axis and decibel scale on      /
// the y-axis                                   /

clear,clc,clf;
w = logspace(-1,3,100);      // Define log scale for w
s = %i*w;                    // Define imaginary s
G = 100../((s-10).*(s-90));  // Define G(s)
y = 20*log10(abs(G));        // Define dB scale for y
plot2d(w,y,5,logflag='ln')   // Plot y=f(w)
xtitle("Bode plot for G(s)=100/((s-10)(s-90))","w,...
   log scale","y, dB scale")
xgrid()                      // Add grid
```

`logspace(-1,3,100)` = "от 10^{-1} до 10^{3} в 100 логарифмически расположенных шагах"

Логарифмическая шкала: график



Гафик был отрегулирован после построения

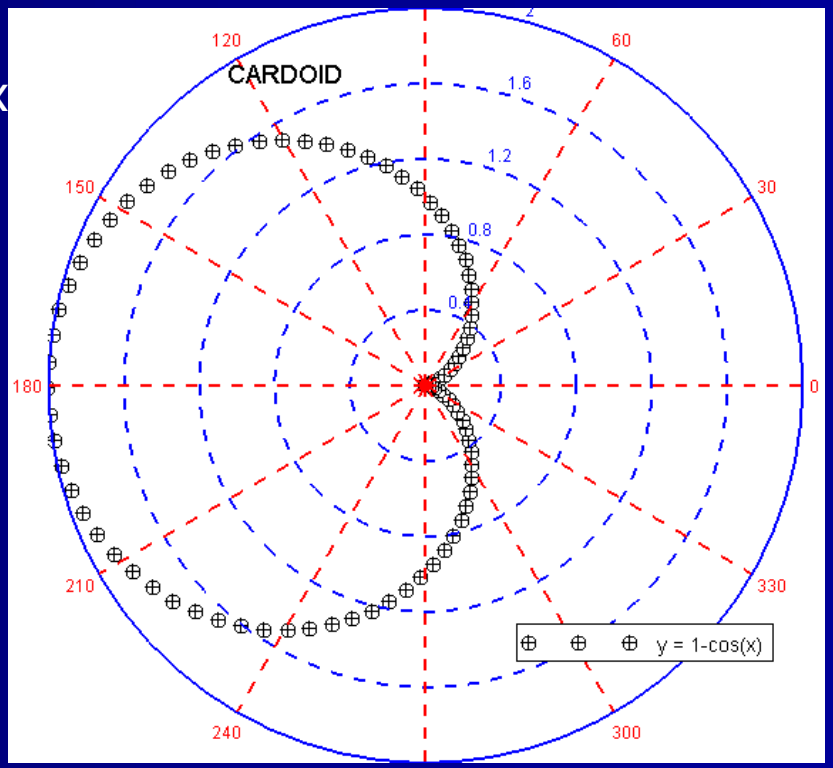
Мы не упоминали ранее аргумент logflag ‘ln’ в plot2d(). Измените ‘ln’ на ‘nn’ (‘ll’ не возможно здесь) и увидите как изменится график (n=normal, l=logarithmic)

Примечание: Scilab имеет специальные функции для графиков Боде, bode(). Смотрите Пример 3-1

Полярные координаты

- Полярные координаты часто используются в некоторых областях машиностроения, например, представить диаграммы лепестков антенны
- Построение в полярных координатах осуществляется командной `polarplot()`
- Демо показывает простой сценарий для кардиоидной `y = 1-cos(x)`, и ее график
- Обратите внимание на маркеры, связанные с аргументом `style= [-3]`
- Редактирование графика занимает много времени для полярных участков

```
//cardioid.sce

// The script plots the cardioid
/
// r = 1 - cos(x), for x = 0...2pi
/
clear,clc,clf;
x = 0:0.07:2*%pi;
polarplot(x,1-cos(x),style=[-3])
legend('y = 1-cos(x)',4)
```



Экспорт графиков

- Scilab графики могут быть экспортированы в различных форматах (PNG, SVG, GIF, Bitmap, и т.д.) для использования в документах
- Чтобы экспортировать, выберите File/Export to... в графическом окне и выбрать файл, а так же желаемый формат
- Альтернативный способ заключается в использовании функции xs2*() которая для PNG принимает форму

```
xs2png(window_number, file_name);
```

- Следующие векторные и растровые форматы **возможны:**

xs2png()	**Экспорт в PNG**
xs2pdf()	**Экспорт в PDF**
xs2svg()	**Экспорт в SVG**
xs2eps()	**Экспорт в EPS**
xs2ps()	**Экспорт в Postscript**
xs2emf()	**Экспорт в EMF (окна)**

xs2fig()	**Экспорт в FIG**
xs2gif()	**Экспорт в GIF**
xs2jpg()	**Экспорт в JPG**
xs2bmp()	**Экспорт в BMP**
xs2ppm()	**Экспорт в PPM**

Обработка (1/12): введение*

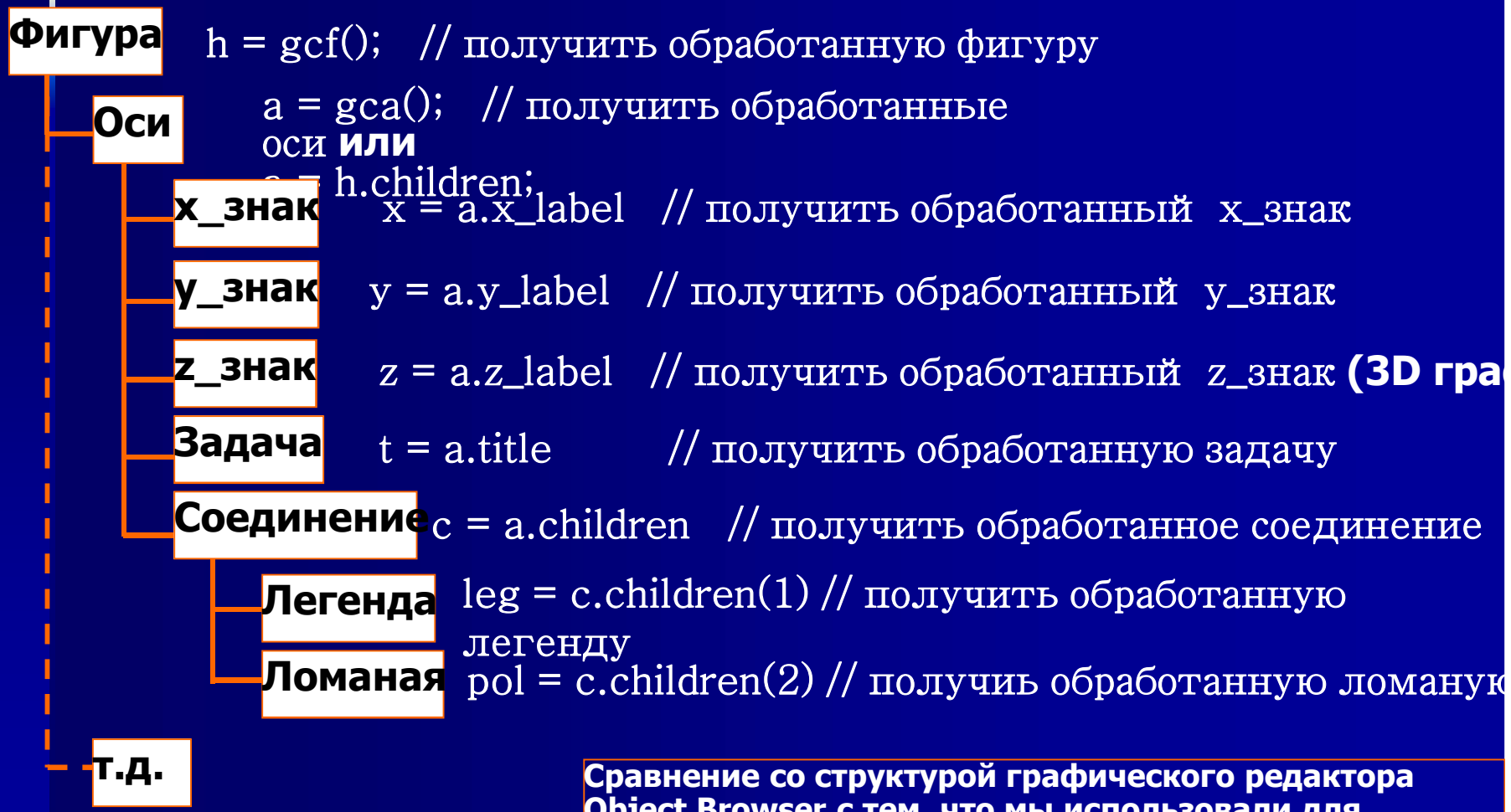
- Обрабоика предназначена для новичков в Scilab. Существуют тексты, данные только в бессвязных некорректных темах. Пользователь остается с опцией "попробовать и плакать"
- Мы ограничимся в этом обсуждении наиболее важными свойствами обработки, направленной на получение базового понимания того, как участки редактируются с обработкой
- Это может помочь посматривать обработку в качестве альтернативы в фигурном редакторе, что мы уже использовали.
- Help Browser обсуждает тему под заголовком graphics_entites. Проверьте так же object_editor

***) Recall the introduction to handles in Chapter 2. Удобные случаи уже использовались в Example 2-3 и когда обсуждались polylines. Это обсуждение основано на Kubitzki: *Grafik, Eigenschaften verwalten in Scilab*, секция 2.4.3 в Champbell др., и Steer: *Scilab Graphics*, 2007.**

Обработка (2/12): введение*

- Графическое Окно построено в виде иерархии объектов. Смотри иерархическое дерево, указанное на следующем слайде, которое также дает типичные команды для каждого объекта
- Верхний объект в окне называется Рисунок. Мы используем функцию `gcf()` , получить текущую фигуру, что бы повлиять на окно, всплывающее на экране. Это делается с обработкой `f = gcf()`
- Рисунок имеет ребенка под названием оси, который в свою очередь имеет несколько детей, и они снова могут иметь собственных детей. Оси имеют название для функции `gca()`, получить текущие оси. Обработка в случае `a = gca()`, и альтернатива `a = f.children` всегда работает
- Обратите внимание на соединение, в котором важны дети Label и Polyline. Последнее относится к фактической графе, что мы наносим
- Рисунок может иметь других детей рядом с осями. Scilab создает их, когда мы используем определенные команды

Обработка (3/12): Основная графическая иерархия

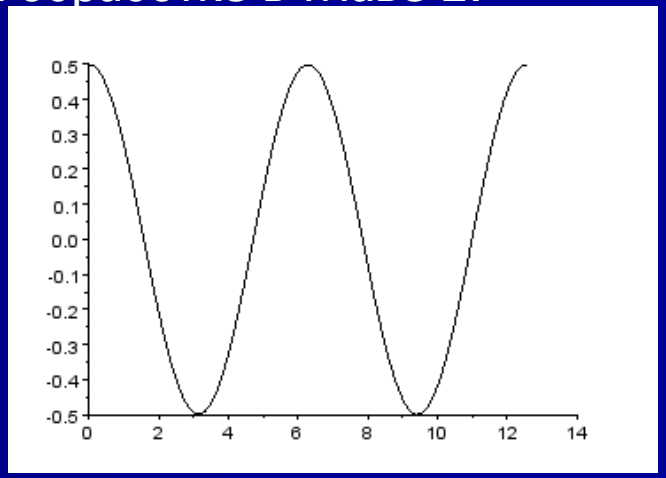


Сравнение со структурой графического редактора Object Browser с тем, что мы использовали для редактирования графики

Обработка (4/12): демо, начальная точка

В качестве первого упражнения, давайте начнем с сценария, который был использован во введении к обработке в главе 2:

```
// handles_demo1.sce

// Basic script to demonstrate handles /

x = linspace(0, 4*%pi, 100);
plot2d(x, 0.5*cos(x))
```



Извлечение из урока:

1) Вы должны быть систематичными при работе с обработкой
2) Существующая литература не всегда правильно. Например, метод, предложенный Steer для изменения оси клещей & знаков просто не работает

Обработка (5/12): демо, за сценой

```
-->gca()

 ans  =


Handle of type "Axes" with properties:

==================================

parent: Figure

children: "Compound“
…
```

Когда мы вызываем обработку осей на консоли, это оказывается очень долго. В верхней части списка мы находим, что оси имеет ребенка соединение

```
-->gce()

 ans  =

 Handle of type "Compound" with properties:

====================================

parent: Axes

children: "Polyline"

visible = "on"

user_data = []
```

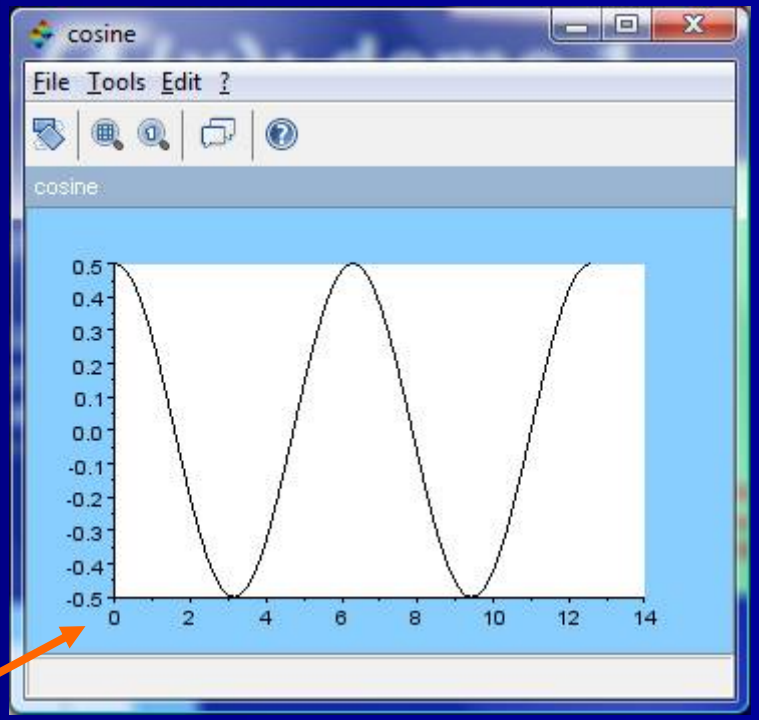
Проверка с `gce()` показывает, что соединение в свою очередь имеет ребенка ломаную линию. Это соответствует иерархии, что мы видели на рисунке редактора

Обработка (6/12): демо, шаг 1

Сначала мы определим некоторые изменения в окне:

- отрегулируйте размер окна
- добавьте цвет фона
- дайте окну имя

```
// handles_demo1.sce

// Basic script to demonstrate handles /

clear,clc,clf;

x = linspace(0, 4*%pi, 100);
plot2d(x, 0.5*cos(x))

f=gcf();    // Get Figure (window) handle
  f.figure_size = [500,400];  // Adjust window size
  f.background = 12;          // Add background color
  f.figure_name= "cosine";  // Name window
```



Проверьте детали о figure_properties **в** Help Browser

Обработка (7/12): демо, шаг 2

В этом шаге мы

- проходите лестницы иерархии
- измените график

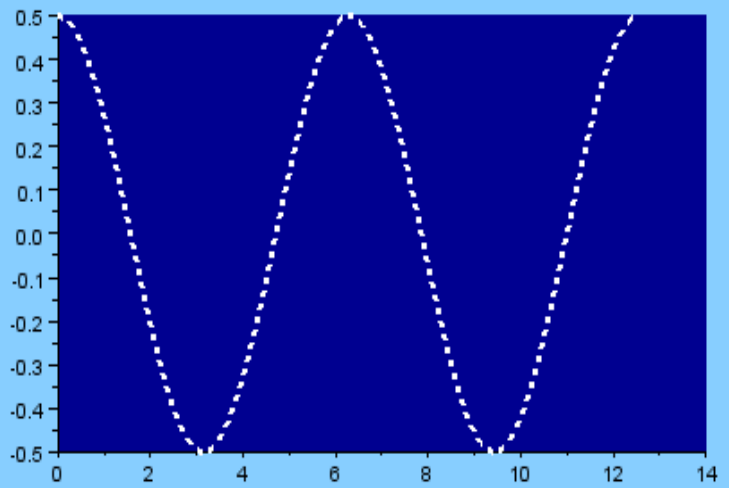
Добавьте эти строки в сценарий (вы можете снять сцену четкости `p1` и писать `c.children.foreground` ... и т.д.)

Обратите внимание, что мы движемся вниз по лестнице иерархии: Рисунок -> оси -> Соединение -> ломаная

Проверьте для детального познания `polyline_properties` в Help Browser

Измените `p1.line_style` **на** `p1.polyline_style` **чтобы получить другой график**

```
a=gca();          // Get Axes handle
  a.background = 9;   // Change background
c=a.children;     // Get compound handle
p1=c.children;    // Get polyline (plot) handle
  p1.foreground = 8;   // Change line color
  p1.line_style = 7;     // Change line style
  p1.thickness = 3;     // Line thickness
```



Обработка (8/12): демо, шаг 3

Как было показано ранее, обработка Entity была довольно пустой. Нам нужно добавить ярлыки, которые можно редактировать. Для этого мы добавляем следующую команду в сценарий:

```
xtitle('COSINE PLOT',...
    'X-axis','Y-axis');
```

А теперь обработка Entity претерпела драматические изменения (это только начало списка)

```
-->gce()
 ans  =

Handle of type "Axes" with properties:
======================================
parent: Figure
children: "Compound"

visible = "on"
axes_visible = ["on","on","on"]
axes_reverse = ["off","off","off"]
grid = [-1,-1]
grid_position = "background"
x_location = "bottom"
y_location = "left"
title: "Label"
x_label: "Label"
y_label: "Label"
z_label: "Label"
auto_ticks = ["on","on","on"]
x_ticks.locations = [0;2;4;6;8;10;12;14]
y_ticks.locations = matrix 11x1
z_ticks.locations = []
x_ticks.labels = ["0";"2";"4";"6";"8";"10";"12";"14"]
y_ticks.labels = matrix 11x1
z_ticks.labels = []
```

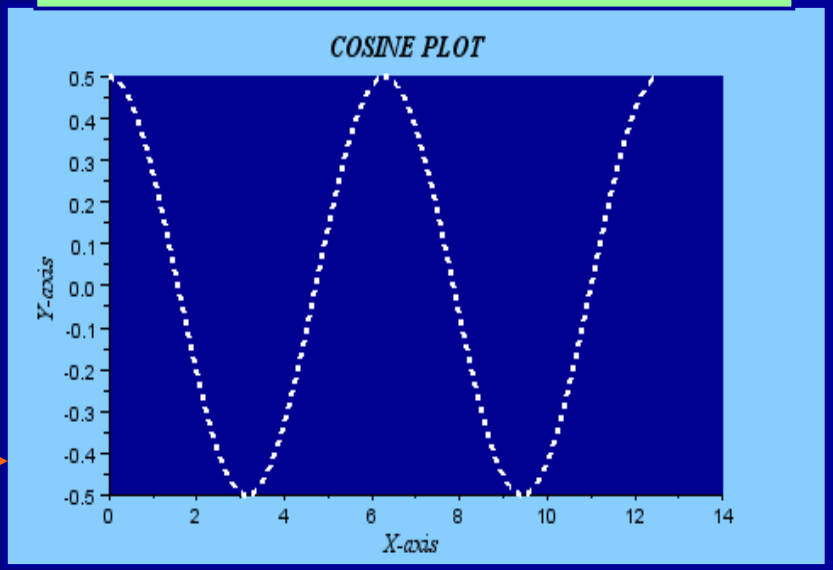
Обработка (9/12): демо, шаг 4

Название и осевые метки были добавлены, следующий шаг заключается в их редактировании
В каждом случае мы должны сначала вызвать соответствующую обработку (или пропустить этот этап, написав команды редактирования в виде a.title.font_style ... и т.д.), что изменит свойства обработки

```
xtitle('COSINE PLOT',...    // Add title & labels
    'X-axis','Y-axis');
t=a.title;                  // Get title handle
  t.font_style = 5;          // Times bold, italic
  t.font_size = 3;           // Increase font size
xL=a.x_label;               // Get x_label handle
  xL.font_style = 5;         // Times bold, italic
  xL.font_size = 2;          // Increase font size
yL=a.y_label;               // Get y_label handle
  yL.font_style = 5;         // Times bold, italic
  yL.font_size = 2;          // Increase font size
```

Используйте Help Browser для детального изучения label_properties

Сюжет не совсем красивый, но мы добавим сетку и редактируем оси отметок и знаков



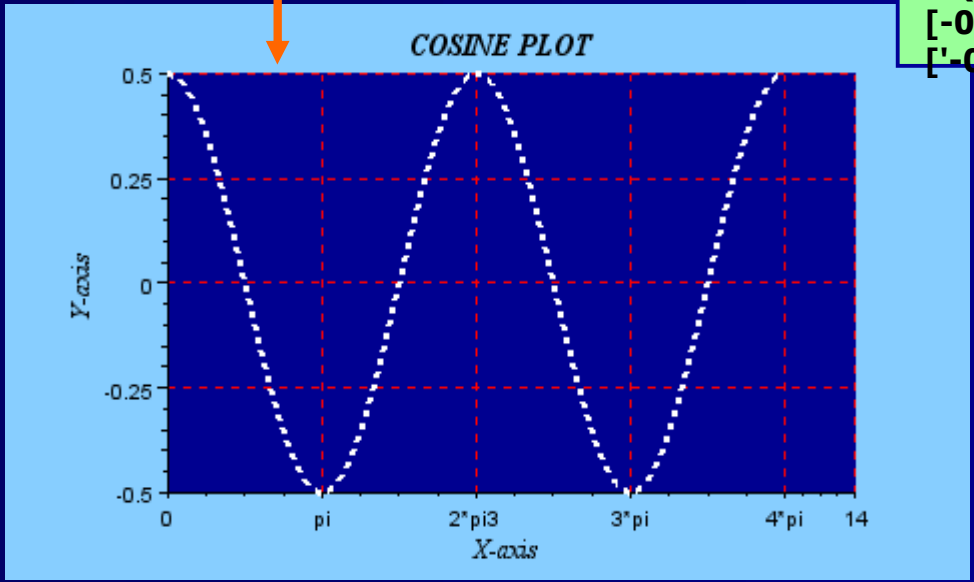
Обработка (10/12): демо, шаг 5

Добавить сетку

Изменить x-ось отметки и знаки

Изменить y-ось отметки и знаки

Конечный график:




```
xgrid(5);      // Add grid

// Change x/y ticks & marks:
a.x_ticks =
tlist(['ticks','locations','labels'],...
   [0,%pi,2*%pi,3*%pi,4*%pi,14],...
   ['0','pi','2*pi3','3*pi','4*pi','14']);
a.y_ticks =
tlist(['ticks','locations','labels'],...
   [-0.5,-0.25,0,0.25,0.5],...
   ['-0.5','-0.25','0','0.25','0.5']);
```

Примечание: Были проблемы с отметками и знаками. Только работал представленный синтаксис

Обработка (11/12): коментарии (1/2)

- С обработкой мы должны соблюдать порядок команд Scilab. Например, сценарий следующего типа вызывает сообщение об ошибке от Scilab:

```
plot(…);
legend(“alpha”, “beta”);
……
a=gca();
a.children(1).foreground=5;
……
```

```
    !--error 15
Submatrix incorrectly defined.

at line       6 of function %h_get called by :
at line      16 of function generic_i_h called by :
at line       2 of function %s_i_h called by :
children(1).foreground = 5;  // Sum pattern re
at line      68 of exec file called by :
opulse_a-pattern.sce', -1
```

- Сообщение об ошибке путает, ссылаясь на подматрицы
- Настоящая причина в том, что мы пытаемся изменить цвет проложенного графика после того, как была объявлена легенда. Scilab не может вернуться назад к легенде и внесите изменения. Команда легенды должна поступить после привязки обработки. Но есть исключения ....

Обработка (12/12): коментарии (2/2)

- Команды обрабокти действительны valid только на специальных уровнях (рисунок, оси, и т.д.). Help/axes_properties дает некоторые намеки но большинство из вас пытаются и плачут и получают сообщения об ошибке

```
!--error 999
This object has no auto_clear property.
at line       4 of function generic_i_h called by
:
at line       2 of function %c_i_h called by :
e2.auto_clear = "on";at line
71 of exec file called by :
examples\planet_moon1.sce', -1
```

- Scilab имеет скрытые повестки дня, когда дело доходит до обработки. Например, нумерация ломаной работает довольно странным образом ...
- Визуальный издание с обработкой, несомненно, улучшает внешний вид фигуры, но является метод "излишним?" Объем кода, необходимого для редактирования сюжета может быть больше того, что используется для создания фактического участка
- Мы должны учитывать, что время = деньги. Важно то, чтобы придумать сценарий, который "соответствует целевому назначению." Остальное роскошь
- Можно изменить настройки по умолчанию Scilab, но информация по данному вопросу является трудно доступной (Kubitzki обсуждает это кратко)

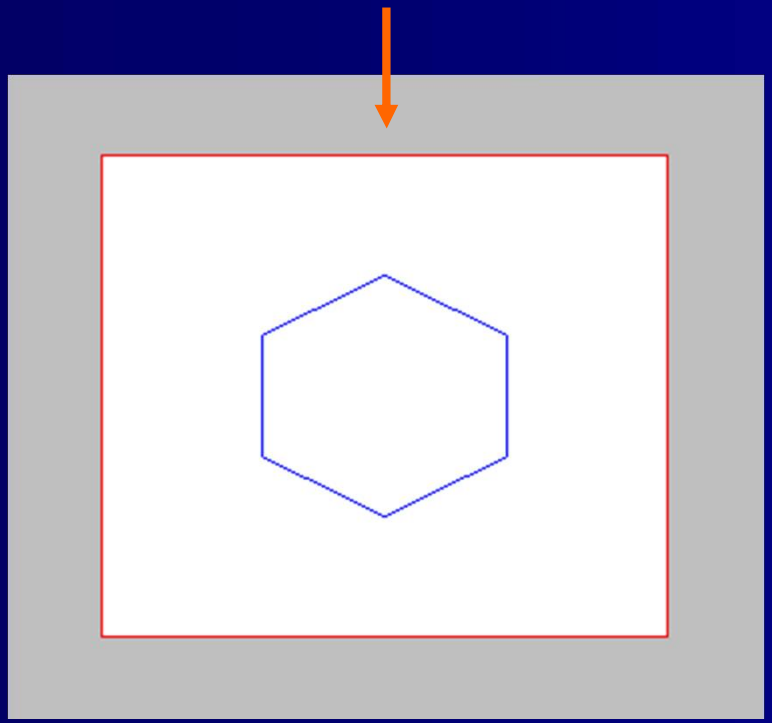
Ломаная (1/3): xpoly(), сценарий

- Это попытка увидеть, насколько хорошо мы можем работать без обычных сюжетных функций
- Сравните пример xpoly(), данный в Help и используемую устаревшую функцию xset()
- Функция xpoly() рисует ломаную линию; замкнутая ломаная получается, если числовой аргумент из xpoly() >0
- Обратите внимание на e.parent.... определение, которое ссылается на одну ступень повыше в иерархии от рисунка
- С e.children.... мы двигаемся на одну ступень в иерархии вниз

```
// xpoly.sce

// Attempt to plot a hexagon with xpoly() & edit   /
// with handles. Causes erroneous behavior in      /
// Scilab. The script must be closed to get rid of  /
// the grey background color                        /

clear,clc,clf;

x = sin(2*%pi*(0:5)/6);       // Define hexagon
y = cos(2*%pi*(0:5)/6);       // - " –
xpoly(x,y,'lines',1);         // Draw polygone

e=gca();                      // Get Axes handle
  e.parent.background =...      // Get Figure handle
    color('grey');              // & set background
  e.box='on';                   // Switch frame on
  e.foreground=5;               // Red frame color
  e.data_bounds=[-2,-2;2,2];    // Frame size
  e.children.foreground = 2;    // Blue graph color
```

Ломаная (2/3): xpoly(), график и обсуждение

И это многоугольник, что мы создали:



Неотредактированный шестиугольник также можно сделать с помощью следующего сценария:

```
x = sin(2*%pi*(0:6)/6);
y = cos(2*%pi*(0:6)/6);
plot2d(x,y,strf='011',rect=[-2,-2,2,2])
```

Это остается открытым, если мы делаем небольшое изменение аргументов

```
x = sin(2*%pi*(0:5)/6);
y = cos(2*%pi*(0:5)/6);
plot2d(x,y,strf='011',rect=[-2,-2,2,2])
```

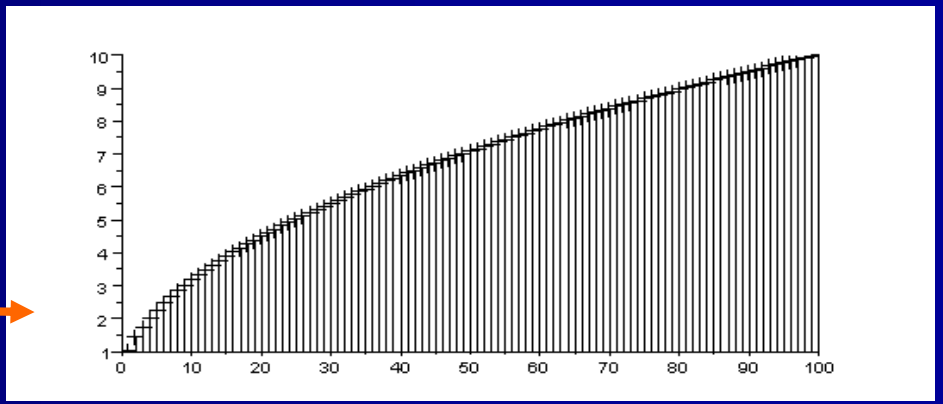
Ломаная (3/3): xpoly(), обучающий урок

- Scilab показал неожиданное поведение с помощью этого сценария:
  - Цвет фона может стать черным с командой `e=gce(); e.parent.background=34` . `gcf()` обработка показала, что была настроена `background=-2` и команда обработки не имела никакого эффекта. The определения (‘серый ’) чувствует себя более стабильно, чем его числовой коллега
  - Графическое Окно не всегда можно изменить, когда сценарий был изменен и выполнен. Задний фон остался серым даже если `e.parent.background=color(‘grey’)` был удален. При переключении между двумя сценариями на редакторе, цвет фона был экспортирован на второй сценарий. Сценарий должен был быть закрыт, чтобы избавиться от серого цвета
  - Я не нашел способ добавления отметок и знаков в коробку. Осевая обработка `gca()` показала как это определено, но по некоторым причинам они подавляются. `Help/axes_properties` не дает никакого объяснения
- Извлечение из урока: не преувеличивайте, в какой степени вы используете обычные графические функции (`plot()`, `plot2d()`) для команд обработки

Ловушки в програмировании: не забудьте clf;

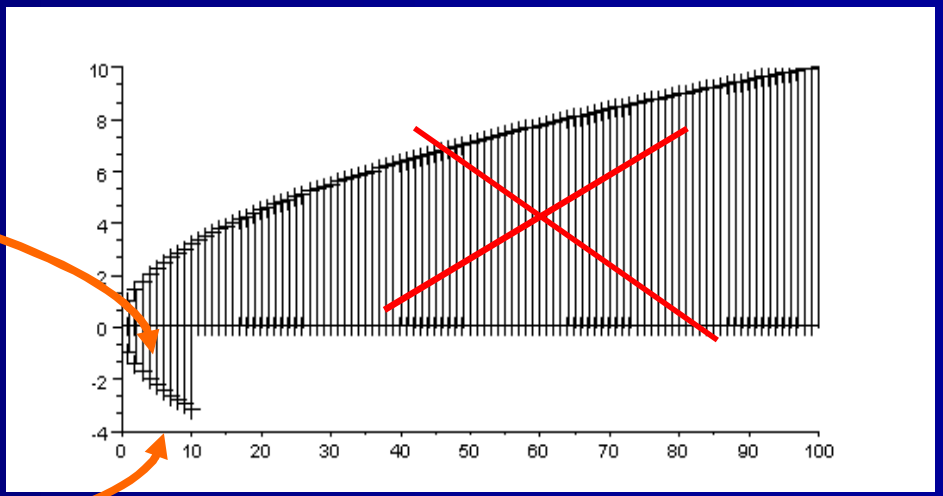
Измените и верните

```
// ptifalls_1.sce

// Clear commands   /

K = 100; a = 0; b = 0;
x = zeros(1,K); y =
zeros(1,K);
for k = 1:K
  x(k) = a+k;
  y(k) = b+k^(0.5);
end
plot2d3(x,y,style=-1)
```




```
// ptifalls_1.sce

// Clear commands   /

K = 10; a = 0; b = 0;
x = zeros(1,K); y =
zeros(1,K);
for k = 1:K
  x(k) = a+k;
  y(k) = b+k^(0.5);
end
plot2d3(x,y,style=-1)
```

Суперпозиция графиков без команды clf

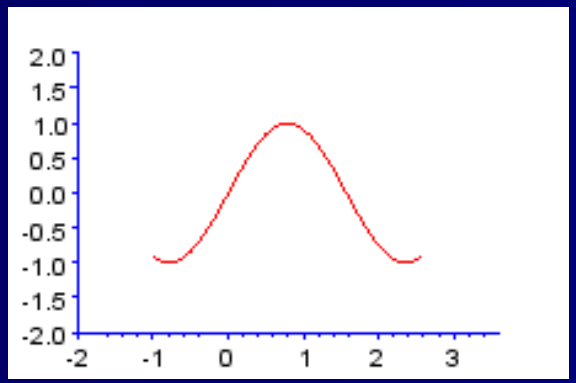


Что делась с xset()?

- Примеры в литературе Scilab и в блогах дискуссий часто используют функцию xset(). Она удобна, но уже устаревшая, так что же нам делать?
- Помощь в браузере рекомендует использовать графические объекты, представленные вместо (set(), get(), команды обработки)
- Ниже приведены примеры того, как подставить xset(). Обратите внимание, что xset() работает на текущем уровне Entity и дает синий цвет осей, не красный график

1. Первоначальный сценарий с xset()

```
x=-1:0.1:2.6
plot2d(x,sin(2*x),5,rect=[-2,-2,3.6,2])
xset("color",2)
```



2. Измененный сценарий с командой осевой обработки

```
x=-1:0.1:2.6
plot2d(x,sin(2*x),5,rect=[-2,-
2,3.6,2])
a=gca();
a.foreground=2
```

3. Измененный график с set() и аргументом обработки

```
x=-1:0.1:2.6
plot2d(x,sin(2*x),5,rect=[-2,-2,3.6,2])
a=gca();
set(a,"foreground",2)
```

xset(): практический случай

Пример 6-2 (в прошлом множество примеров) взята из Pinçon. В оригинале содержатся устаревшие команды, в практике `xset()`. Я заменил соманду `xset()` с помощью следующих команд обработки:

Команда xset()		Команды графической обработки
		h=gca()
xset('background',1)	Черный	h.background = 1
xset('color',2)	Заполнение синим	h.foreground = 2
xset('thickness',3)	толщина линии	h.thickness = 3
xset('color',5)	красная граница	h.foreground = 5

Но, откровенно говоря, это может быть мучительно и вы захотите выбросить компьютер в окно. Если это так, то проверьте обработку `gca()`, имеет ли она детей вообще...

Проблемные сообщения об ошибках

- Отладчик Scilab показывает недостатки в сообщениях об ошибках, которы неизменно всплывают до правильных команд
- Вот два сообщения об ошибках, которые я получил:

```
plot2d(x1,y1,style=5)  // Function
                  !--error 999 plot2d:
 first and second arguments have incompatible dimensions.
at line      16 of exec file called by :
   exec("H:/Dr.EW/Writings/Scilab examples/derviative_2.sce");
while executing a callback
```

Реальная проблема в том, что я не использовал оператор точки в уравнении для y1

```
clc();
      !--error 13

Redefining permanent variable.

while executing a callback
```

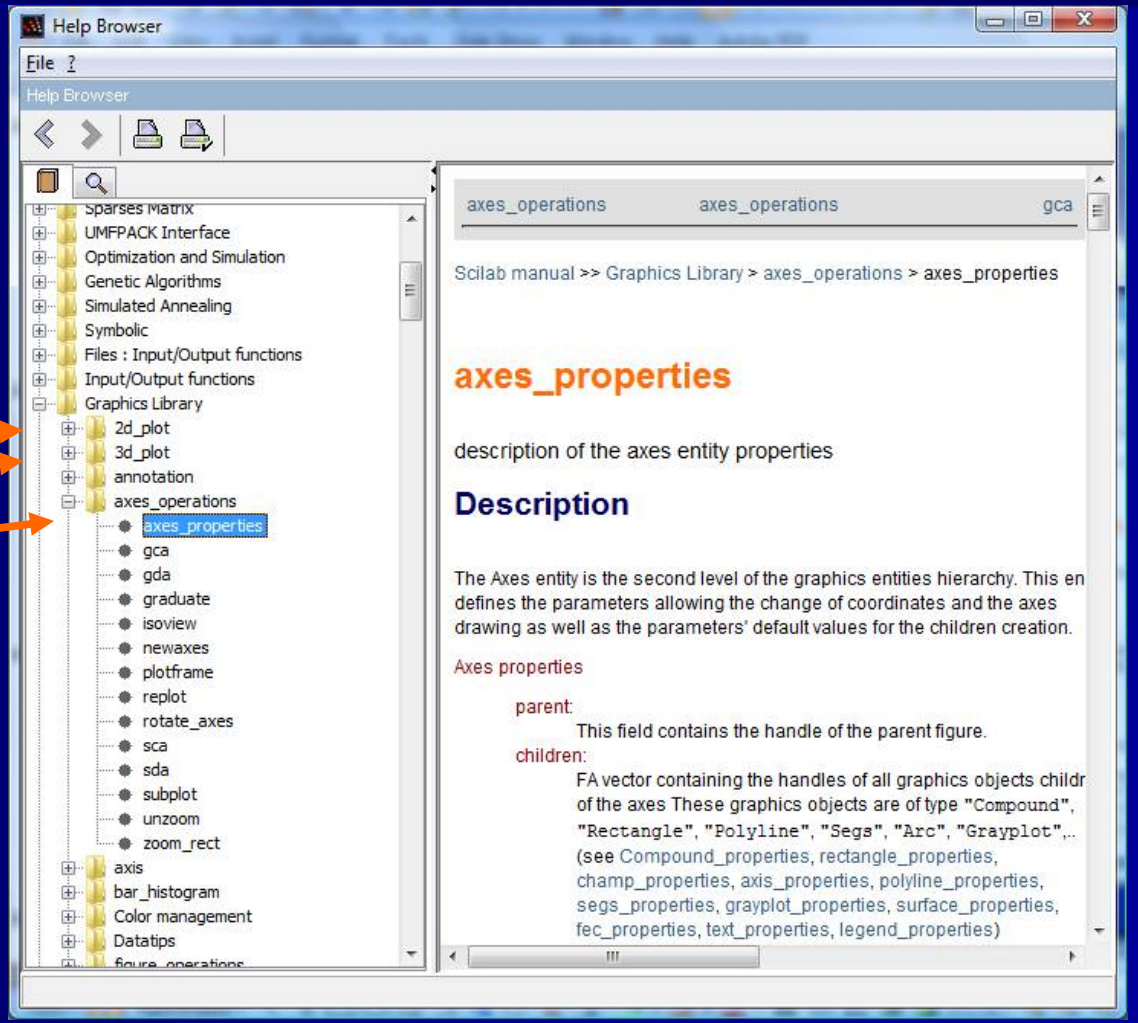
Поддельное предупреждение, Scilab сбился и должен был быть перезагружен, чтобы стереть консоль. Это произошло, когда я использовал две функции Deff () в тандеме

Больше информации о построении

В Help Browser,Нажать: графическая библиотека, а под ней вы найдете информацию о, например
- 2D plots
- 3D plots
- axes_operations/ axes_properties

и др.

Теперь, это время вы получили чтобы ознакомиться с Help Browser



Dr. EW
Johnny Heikell

8. Примеры №3

На построение, обработку, контроль инженерных и определяемых пользователем функций

Пример 3-1: больше контроля инженерными графиками

- Example 2-3 и логарифмическая шкала демо были типичным управлением инженерных задач. Напомним так же страницы на полиномы в Chapter 3
- Здесь мы будем смотреть на примеры с графиками Боде и Найквиста, диаграмма Николса (черная диаграмма), и график Эванса
- Первые случаи использования функций передачи второго порядка

$$G_2(s) = \frac{s^2 + 20s + 100}{s^2 + 6s + 100} * \frac{s^2 + 3s + 220}{s^2 + 25s + 225}$$

- Корневой локус Эванса построен для

$$G_4(s) = 352 * \frac{5 + s}{2000s^2 + 200s^3 + 25s^4 + s^5}$$

Пример 3-1: Сценарий

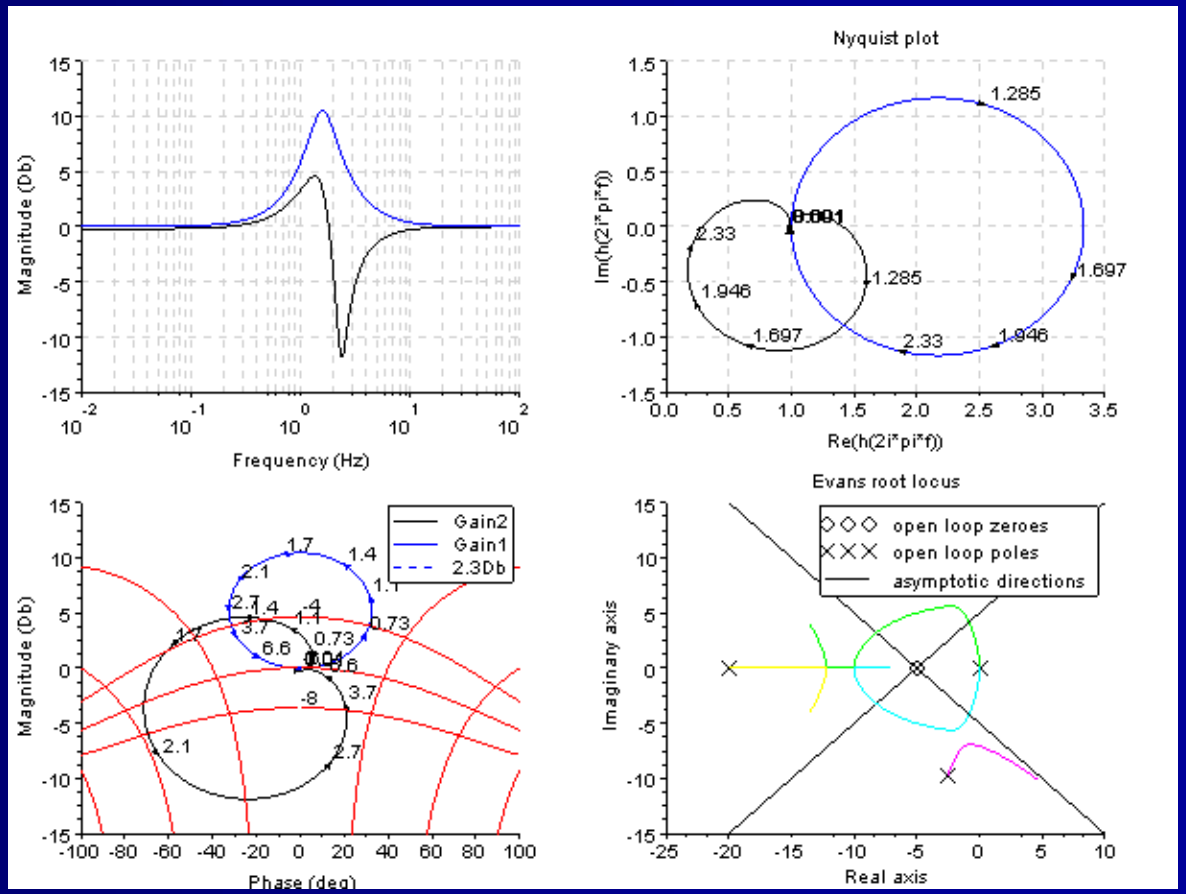
- Первые два уравнения усиления даются как обычные полиномиальные выражения
- Третье уравнение усиления, которое будет использоваться в построении корневого локуса Эванса, определяется через свои корни
- График Боде только для усиления, позже альтернатива `bode()` будет продемонстрирована
- Scilab говорит о диаграмме Блэка и о графике Николса. Пример 3-2 подчеркивает разницу между ними

```
// control_eng.sce

// Plot Bode, Nyquist, Nichols & Black's,  /
// and Evans for defined equations        /

clear,clc,clf;
// Definition of systems:
//-----------------------------
s = poly(0,'s');   // Polynomial seed
Gain1 = syslin('c',(s^2+20*s+100)/(s^2+6*s+100));
Gain2 =
Gain1*syslin('c',(s^2+3*s+220)/(s^2+25*s+225));
Gain3 = poly(-5,'s')/poly([0,0,2000,200,25,1],'s','c');
Gain4 = syslin('c',352*Gain3);

// Bode plot:
//--------------
subplot(221)
gainplot([Gain2;Gain1],0.01,100)  // Magnitude plot

// Nyquist plot:
//------------------
subplot(222)
nyquist([Gain2;Gain1])  // Plot with Re and Im axes

// Nichols chart (Black's diagram + iso-plots):
//-------------------------------------------------------------
subplot(223)
black([Gain2;Gain1],0.01,100,[`Gain2';`Gain1'])
chart([-8 -6 -4],[20 50 80],list(1,0,5))

// Evans root locus:
//-----------------------
subplot(224)
evans(Gain4,100)   // Evans root locus for sys4
```

Пример 3-1: график

График не редактировался, все показанное, является результатом сценария. Обратите внимание на красные кривые на подграфике Боде-Николоса

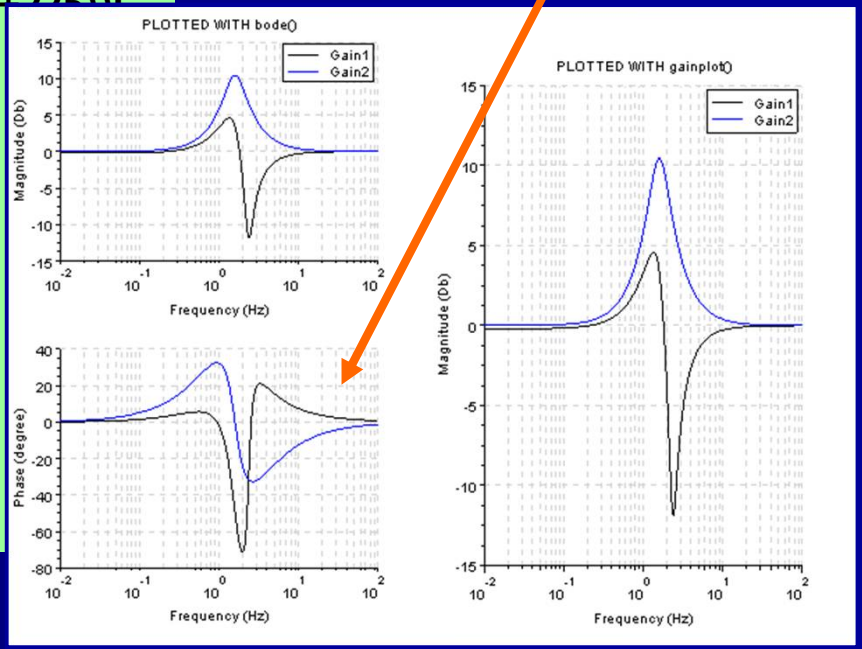
На следующем слайде рассматривается, как работают альтернативные команды Боде



Пример 3-1: альтернативные графические функции Боде

```
// bode_comparison.sce

// Compare the how the bode() and gainplot()   /
// functions operate in Example 7              /

clear,clc,clf;
s = poly(0,'s');   // Polynomial seed
Gain1 = syslin('c',(s^2+20*s+100)/(s^2+6*s+100));
Gain2 =
Gain1*syslin('c',(s^2+3*s+220)/(s^2+25*s+225));

// Plot with the bode() function:
//-------------------------------------
subplot(121)
bode([Gain2;Gain1],0.01,100)
legend(`Gain1',`Gain2')
xtitle('PLOTTED WITH bode()')

// Plot with the gainplot() function:
//-----------------------------------------
subplot(122)
gainplot([Gain2;Gain1],0.01,100)
legend(`Gain1',`Gain2')
xtitle('PLOTTED WITH gainplot()')
```

Этот пример демонстрирует функции bode() и gainplot() при работе на ранних Gain1 и Gain2 выражениях. bode() строит так же фазы



Пример 3-1: коментарии

- Сценарий был изменен после копирования из Руководства Scilab Group User's и вставлен в редактор. При копировании вставки, редактор имеет тенденцию интерпретировать цитирование марок (‘c’,‘s’, etc.) ошибочно, и они должны быть исправлены вручную
- Scilab строг с аргументами для полиномиальных выражений. Если, например, "C" осталось из выражения poly([0,0,2000,200,25,1],’s’,’c’), он будет переведен в $10000000s^2 - 10455000s^3 + 455225s^4 - 2226s^5 + s^6$. Будьте осторожны!
- Существует преимущество в использовании самодокументируемыми выражения, назвав полиномы Gain1, Gain2, и т.д.
- Отдельный демо график Bode показывает, что функция `bode()` имеет преемущество в предоставлении и так же фазу интересной системы
- Разница между диаграммы Блэка и графика Николса будет показана в примере 3-2

Пример 3-2: Блек и Николс

Этот пример, адаптирован от учебника Povy стр. 78, показывает, что команда chart() добавляется к диаграмме Блека

Первый аргумент вектора chart() определяет увеличение кривой для печати

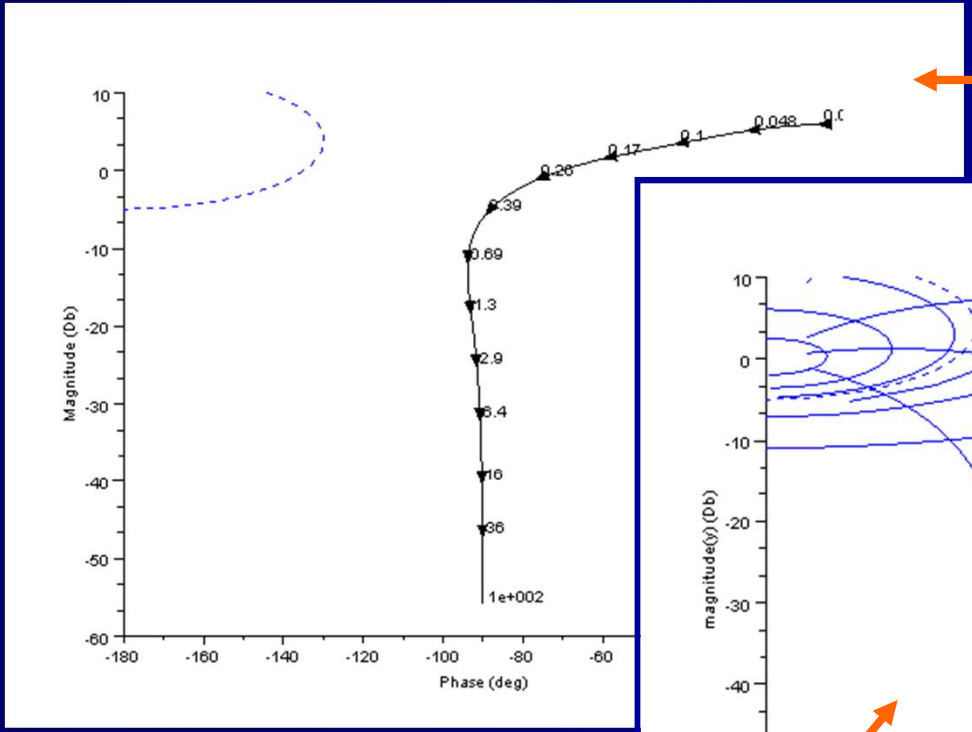
Второй аргумент определяет изо-фазовые кривые

list() определяет почтроение свойства (последний аргумент не имеет никакого эффекта)

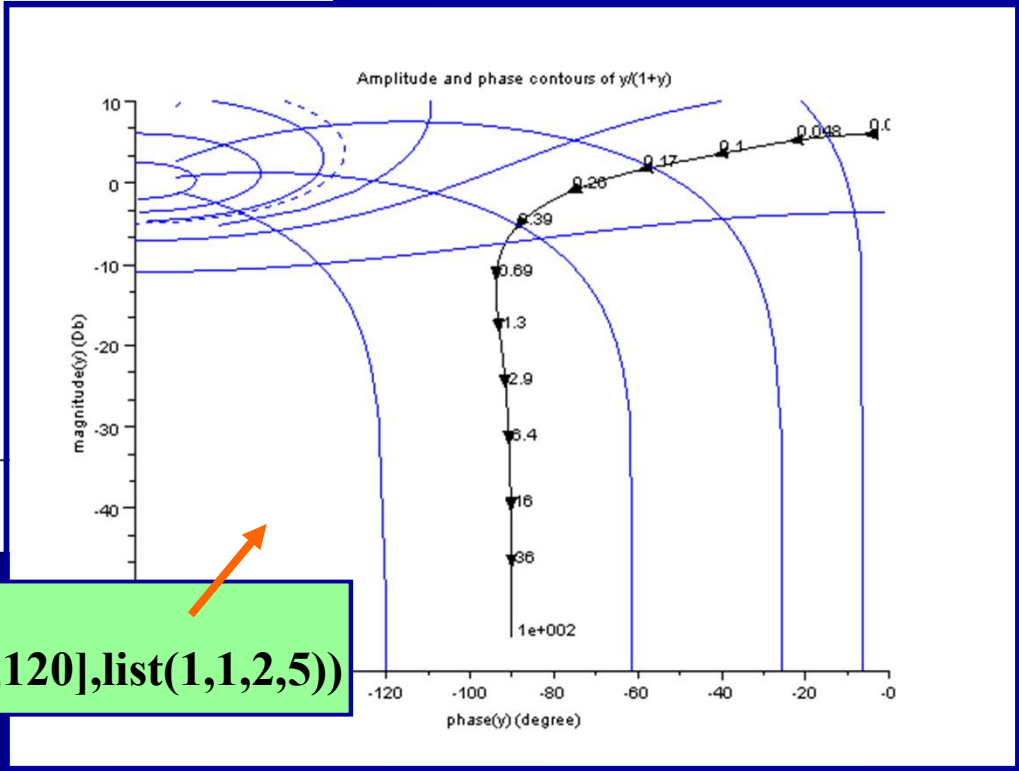
Используйте Help для получения деталий

```
// black_nichols.sce

// Demonstration of black() and  /
// chart() functions                    /

clear,clc,clf;

s = %s;
Gain = (2+3*s+s^2)/(1+3*s+2.5*s^2+s^3);
system = syslin('c',Gain);

black(system,.01,100)  // Plot Black's diagram
chart([-8,-
2,.5,3,6,12],[5,25,60,120],list(1,1,2,5))
                              // chart() adds iso-graphs
```

Пример 3-2: графики



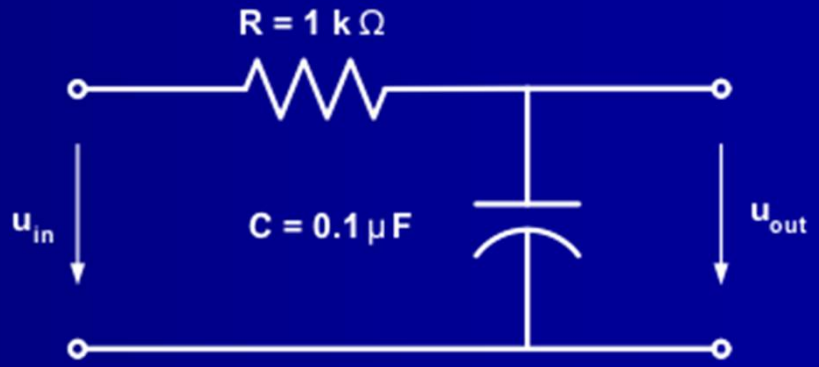
black(sl,.01,100)



black(sl,.01,100)
chart ([-8,-2,.5,3,6,12],[5,25,60,120],list(1,1,2,5))

Пример 3-3: RC схема

- Давайте сделаем график Bode, использующий только базовую теорию цепи и без всякого мусора Лапласа
- Простая схема RC с права (фильтр низких частот перого порядка)
- Задача состоит в том, чтобы построить величину и фазу
- Функция `bode()` не подходит для этого случая, вместе этого мы будем использовать `plot2d()` и определим ее отдельно для амплитуды и фазы



$$G = \frac{u_{out}}{u_{in}} = \frac{1}{1 + i2\pi f RC}$$

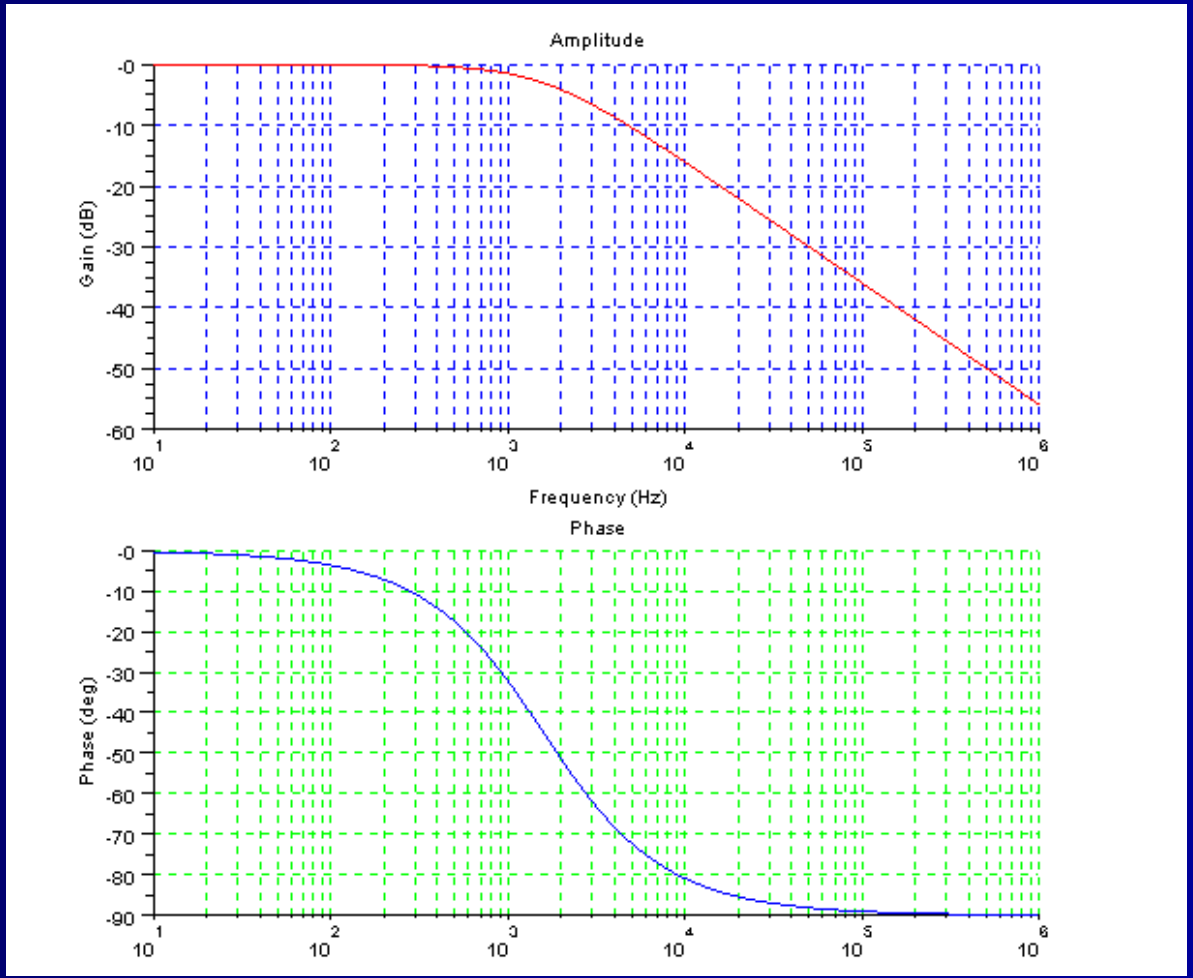
Пример 3-3: сценарий

- Команда logspace(1,6,60) означает начальную точку 10^1, конечную точку 10^6, в 60 шагов и логарифмическую шкалу
- Определение тригонометрической фазы и преобразования в градусы
- Аргумент logflag = 'ln' определяет логарифмическую x-шкалу и линейную (нормальную) y-шкалу
- Различные стили и xgrid() аргументы были использованы чтобы продемонстрировать свои эффекты

```
// bode_RC.sce

// Bode diagram for an RC circuit   /
// (first-order low-pass filter)    /

clear,clc,clf;

R = 1e+3;      // Resistance in ohm
C = 1e-7;      // Capacitance in farad
freq = logspace(1,6,60);  // Frequency range, logarithmic
G = 1 ./ (1 + %i*2*%pi*freq*R*C);  // Transfer function
G_dB = 20*log10(abs(G));           // Logarithmic scale
phase = ((atan(imag(G),real(G)))/(%pi))*180;  // Phase

subplot(211);                      // Amplitude plot
plot2d(freq,G_dB,logflag='ln',style=5)
xgrid(2)                           // Blue grid
xtitle('Amplitude','Frequency (Hz)','Gain (dB)')

subplot(212)                       // Phase plot
plot2d(freq,phase,logflag='ln',style=2)
xgrid(3)                           // Green grid
xtitle('Phase','Frequency (Hz)','Phase (deg)')
```

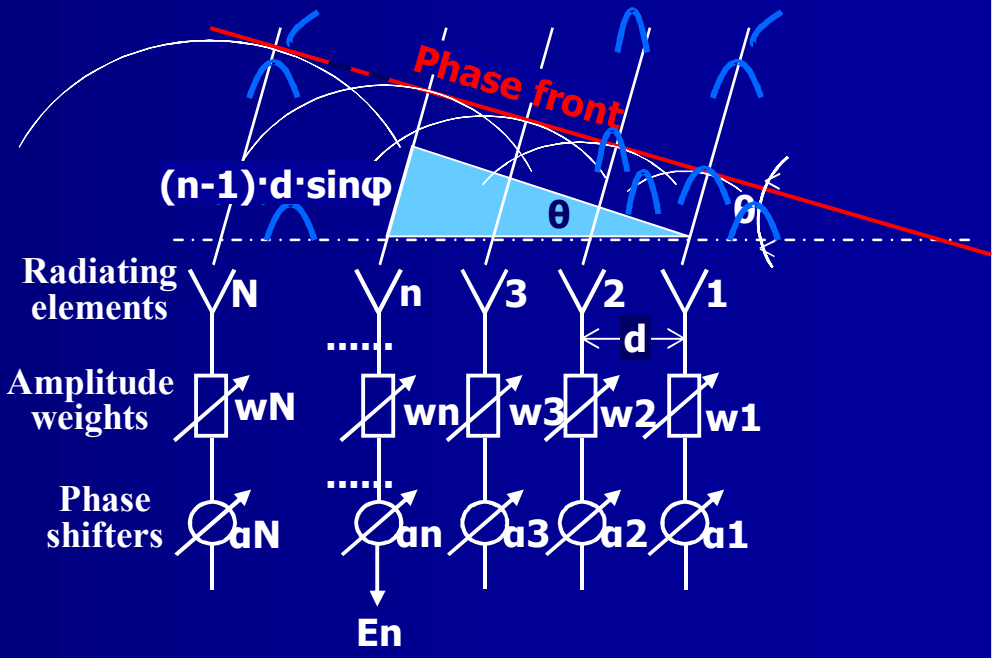
Экс 3-3: plot

Обратите внимание, что для фазы участка метка оси x отсутствует, хотя это было указано. Scilab не повторяет его, поскольку он является таким же, как верхняя. Измен ение заявления SubPlot в (121) и (122) и метка оси X дается для обеих частей.



Пример 3-4: линейная антенная решетка

Задача состоит в том, чтобы исследовать поведение массива, AF (также известен как шаблон интенсивности поля), линейной антенной решетки с N = 10 изотропными излучающими элементами, когда сканируется главный луч на θ = 60 º и элемент расстояние г = 0,45 и 0,55 длины волн
(Для обсуждения антенных решеток, см. Брукнер, Е. (ЭД): Практическая фазированной антенной решетки Systems, Artech House, 1991)



$$|AF| = \left| \frac{\sin\left[N\pi \left(\frac{d}{\lambda}\right)\cdot\sin\theta \right]}{\sin\left[\pi \left(\frac{d}{\lambda}\right)\cdot\sin\theta \right]} \right|$$

Экс 3-4: сканирующий луч

Предыдущее выражение для AF действительно, только когда луч расположен перпендикулярно оси массива (залп случай). Если пучок отклоняется, сканирование угла θ_o должно быть включено в уравнение (В более полной имитации мы должны также включать факторы элементов и взаимную связь между элементами массива)

Scan angle

$$|AF| = \left| \frac{\sin\left[N\pi\left(\frac{d}{\lambda}\right)\cdot(\sin\theta - \sin\theta_o)\right]}{\sin\left[\pi\left(\frac{d}{\lambda}\right)\cdot(\sin\theta - \sin\theta_o)\right]} \right|$$

Scan angle

Экс 3-4: script

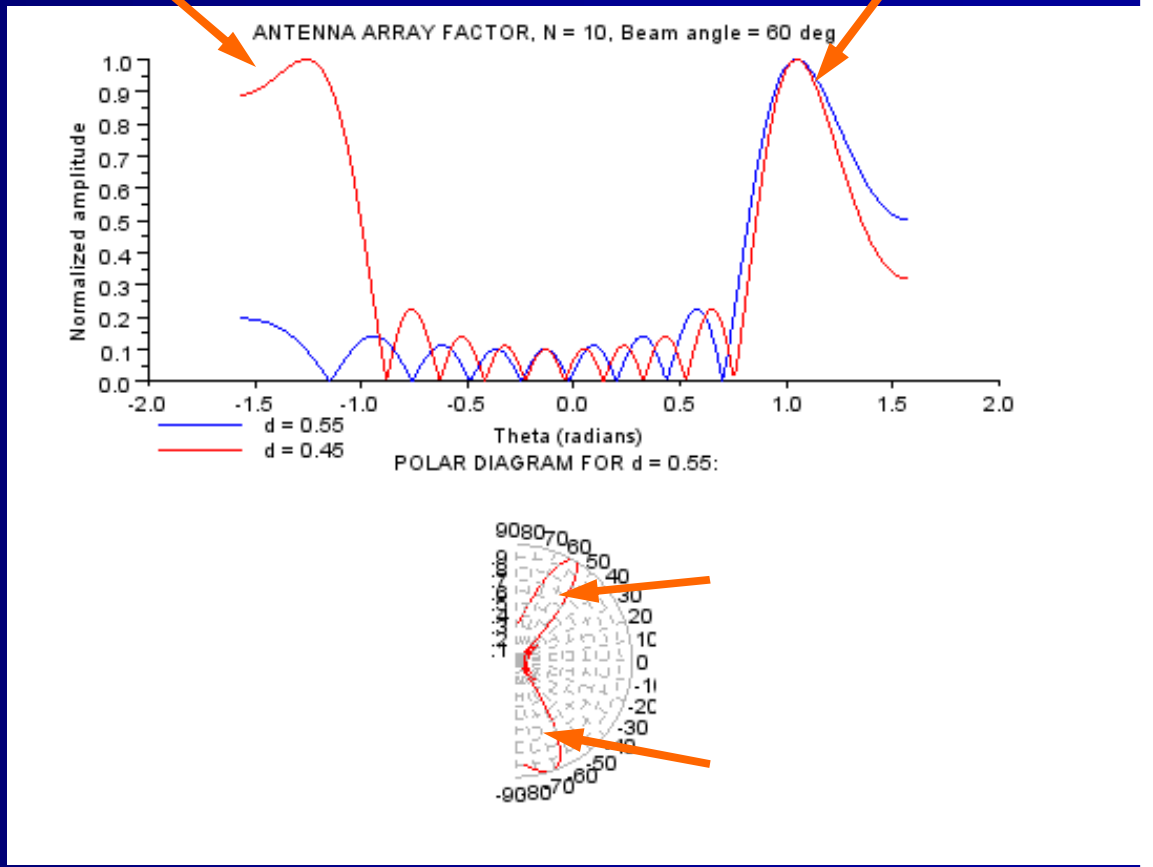
Это script для предыдущего выражения для Af, но он нормализуется (деленное на число N), чтобы сохранить главную ценность луча в единстве. Фазовые функции были определены отдельно, чтобы сократить выражение для AF_norm
Коэффициент массива строится как в линейной, так и в полярной презентации.

```
// array_factor.sce

// ---------------------------------------------------------------------
// Plots the array factor of a linear antenna array with N elemnts
// spaced at d wavelengths, and main beam scanned at +60 degr
// ---------------------------------------------------------------------

clear,clc,clf;

// Variables:
N = 10;         // Number of radiating elements
d1 = 0.45;      // Element spacing in wavelengths
d2 = 0.55;      // Ditto
theta = [-%pi/2:0.01:%pi/2]';    // Half space, +/-90 deg
theta_z = %pi/3;                 // Scan angle

// Define array factors:
f_phase1 = %pi*d1*(sin(theta)-sin(theta_z));    // Phase functio
f_phase2 = %pi*d2*(sin(theta)-sin(theta_z));    // Ditto
AF_norm1 = abs((sin(N*f_phase1)./sin(f_phase1)))/N);
          // Normalized array factor (d=0.45)
AF_norm2 = abs((sin(N*f_phase2)./sin(f_phase2)))/N);
          // Normalized array factor (d=0.55)

// Plot functions:
subplot(211                      // Linear plot (d=0.45,0.55)
plot2d(theta,[AF_norm1,AF_norm2], style=[2,5],...
   leg="d = 0.55@d = 0.45")
xtitle("ANTENNA ARRAY FACTOR, N = 10, Beam angle = 60 deg",
   "Theta (radians)","Normalized amplitude")

subplot(212)                     // Polar diagram (d=0.55)
polarplot(theta,AF_norm2, style=5)
xtitle('POLAR DIAGRAM FOR d = 0.55:')
```

Экс 3-4: plot

Grating lobe

Main lobe



Plot проверяет общее эмпирическое правило, согласно которому расстояние элемент массива должно удовлетворять условию г <λ / 2 или вредные дифракционные максимумы будут отображаться Обратите внимание, что является зеркальным отражением в другом полупространстве, только º в случае ± 90 была построена

Экс 3-4: изменение plot

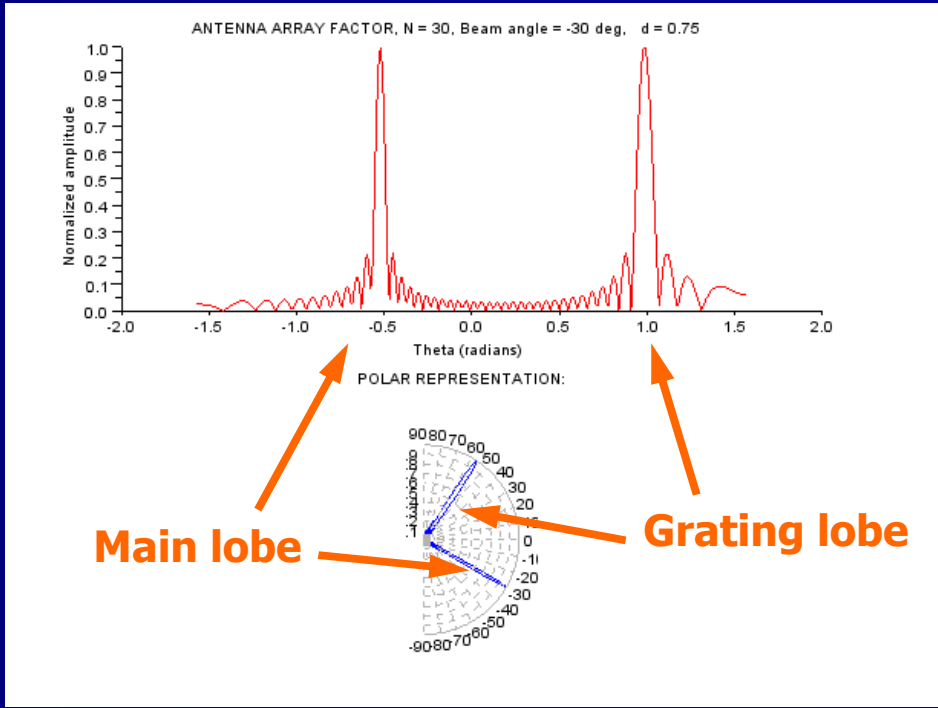
Этот случай показывает, d=0.45. Другие изменения:

- Элемент расстояние г = 0,75

- Номер элемента N = 30

- Угол сканирования = -30 o

Scilab 5.1.1 тенденцию в условиях, как представить только часть полярных координатах, но в то же время увеличить размер, оставив годограф переполненным нелинейными plotами (который не были плохими). Это, кажется



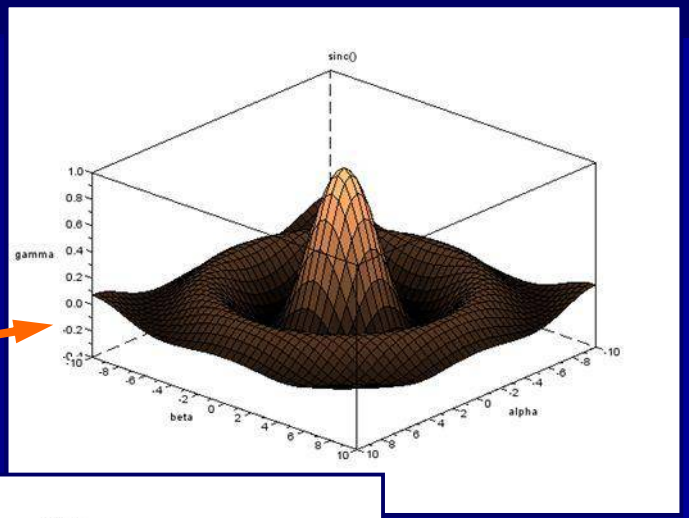
Пример 3-5: 3D sinc

Этот пример взят из Kubitzki, Einführung в Scilab, стр. 41-42, и можно сравнить с тем, который использовали раньше для построения функций 3D SINC Script знакомит с использованием цветовых карт для графических идентификации цвета. Аргумент coppercolormaps () определяет количество цветов на карте; 32-частовстречающаяся цифра Обратите внимание, что color_map работает на рисунке. уровня (F = GCF ()) Здесь мы используем пару drawlater () и drawnow (), чтобы контролировать процесс на участке

```
// sinc_colormap.sce
// Define and plot 3D sic funtion, graphic /
// adjust properties with handles /

clear,clc,clf;
x=linspace(-10,10,50); // Linear space of x
y=x; // Ditto for y

// **** SUBROUTINE sincf(): **** |
//-------------------------------------------
function [z]=sincf(x, y)
  r=sqrt(x.^2+y.^2)+%eps;  // Auxiliary computation
  z=sin(r)./r;                    // Amplitude
endfunction

// **** MAIN, Compute sinc function: **** |
//-----------------------------------------------------------
w=feval(x,y,sincf);  // Evaluate with SUBROUTINE sincf()

// Plotting & plot control:
//-------------------------------
drawlater();          // Suppress plotting until ready
  plot3d(x,y,w);     // (Suppressed) plot function
  f=gcf();              // Get Figure handle
  f.color_map = coppercolormap(32);   // Set color table
  h=gca();                          // Get Axes handles
  h.rotation_angles=[87,42];   // Set angle of observation
  h.children.color_flag=1;        // Use current color table
  xtitle('sinc()','X','Y','Z');          // Title & legend
drawnow();                          // Plot now
```

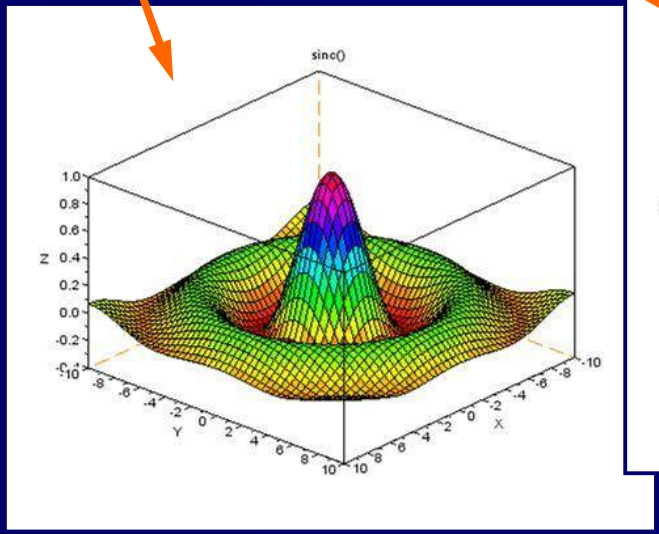
Экс 3-5: 3D sinc, plots и комментарии

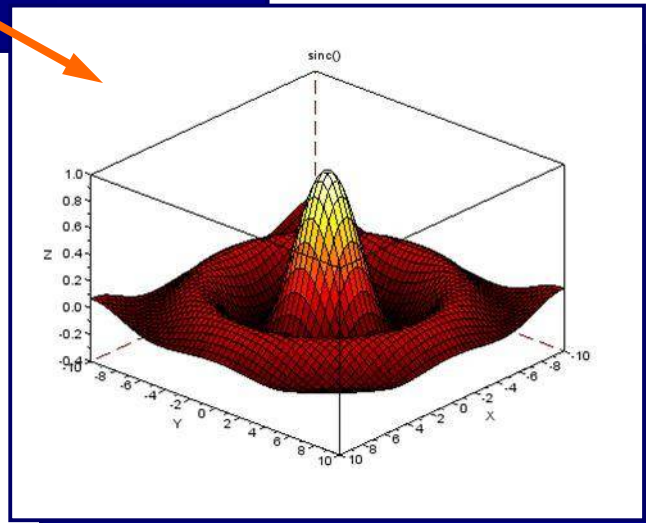
Scilab имеет многочисленные альтернативы цветов карты, которые позволяют изменять цвет 3D участка, например того, который показан здесь.



hsvcolormap()

hotcolormap()





Пример 3-6: Lissajousfigures, задача

Задача состоит в том, чтобы написать script, который генерирует фигуры Lissajou и редактирует фигуру с ручками

Фигуры Lissajou знакомы всем, кто работал с помощью осциллографа в средней школе физической лаборатории

Математически фигуры Lissajou -это график системе параметрических уравнений типа:

x = · (грех (ωt) + φ)

y = B · грех (ωt)

Мы будем строить две фигуры в одном окне, сочетание

sin(x) & cos(3x) and

sin(1.5x) & 0.5*cos(1.5x)

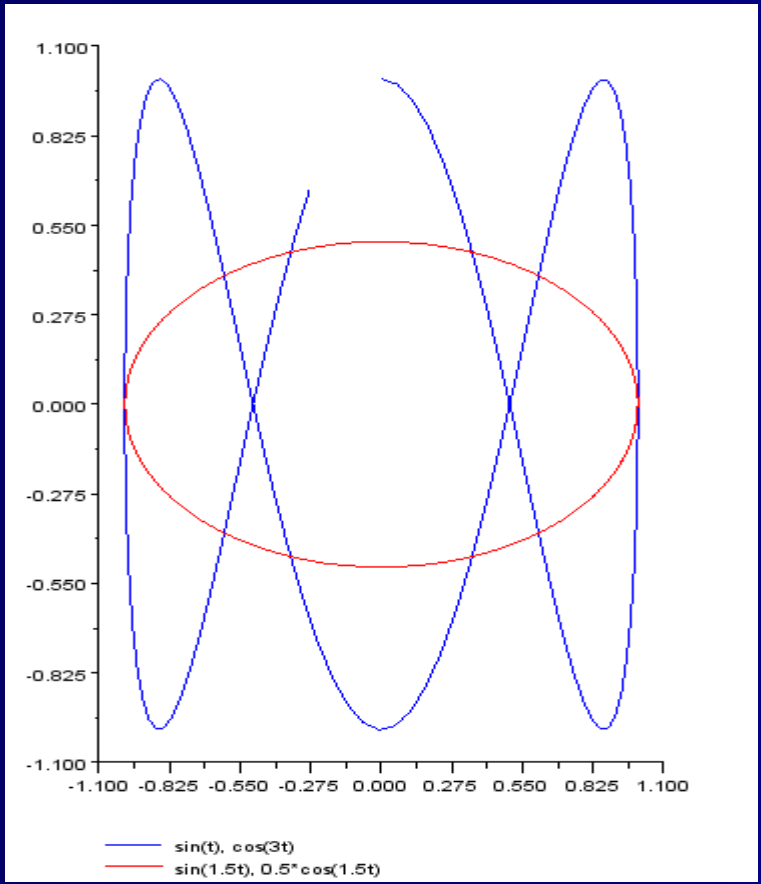
Для сравнения мы сначала сделаем основной plot с plot2d (), а затем изменим фигуру с ручками

Экс 3-6: Lissajousfigures, script 1

- Синус и косинус функции сгруппированы в матрицах
- Plot2d () Аргумент [2,5] определяет графа цвета
- Аргумент нога определяет легенду
- Аргумент nax определяет осb подразделения
- Аргумент прямоугольник определяет расширение осей X и Y.

```
// handles_demo2-1.sce
// Two Lissajous figures, sin(t) & cos(3t) and       /
// sin(1.5t) & 0.5*cos(1.5t), with plot definitions  /
// given by arguments of the plot2d() function       /

clear,clc,clf;

// Plot Lissajous figures:
//-------------------------------
t=linspace(0,6,100)';
sines = [sin(t) sin(1.5*t)];
cosines = [cos(3*t) 0.5*cos(1.5*t)];
plot2d(sines, cosines, [2,5], ...
  leg='sin(t), cos(3t)@sin(1.5t), 0.5*cos(1.5t)',...
  nax=[1,9,1,9], rect=[-1.1,-1.1,1.1,1.1])
```

Экс 3-6: Lissajousfigures, plot 1



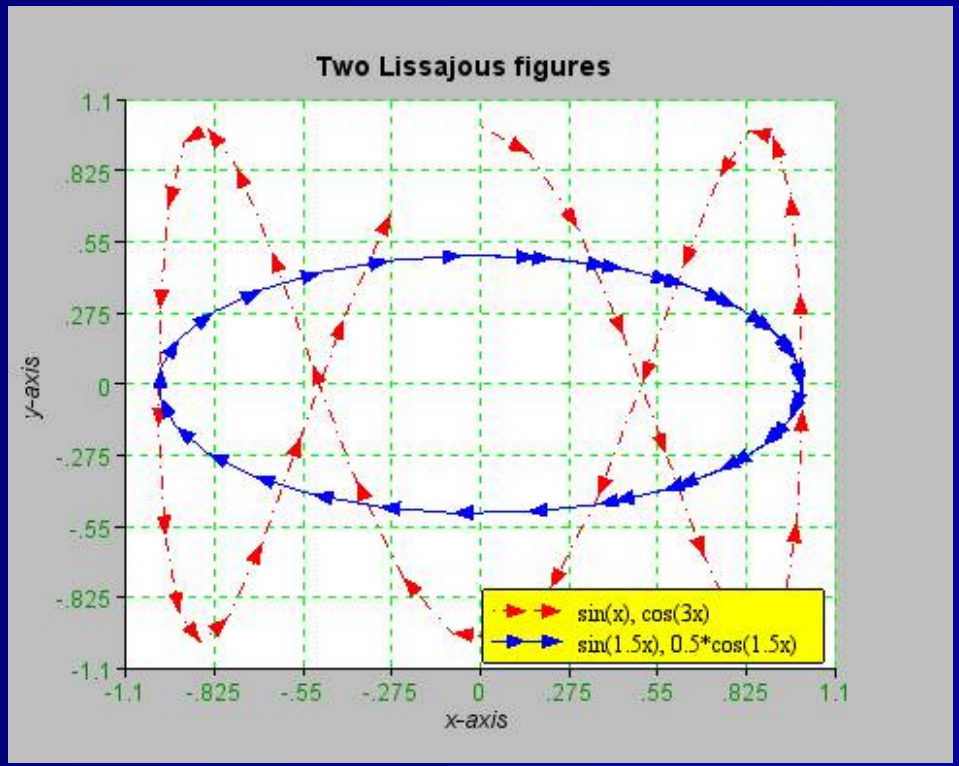
На рисунке определяется **sin(t),** потому что **cos(3t)** не совсем закончен (его выдвижения определяется аргумента 6 в LINSPACE ())

На втором рисунке, **sin(1.5t), 0.5*cos(1.5t)** уже на своем втором цикле. Эллипс превращается в окружность, если мы изменить амплитуду косинуса 1

Обратите внимание на то, что plot2d () сочетает в себе аргументы синусов и косинусов. Элемент-от-элемента.

Экс 3-6: Lissajousfigures, plot 2

Это plot, который был изменен с помощью ручки. * Script представлен на следующих четырех слайдах
Основные модификации:
- Обе фигуры Lissajou являются стрелками, одна строка штрихпунктирной
- Название и оси отметки были добавлены и отредактированы
- Цвет фона была добавлен
- Легенда коробка была введена в нижнем правом углу, текст редактируется и коробка цвета добавляется
- Сетка была добавлена и отредактирована



***) Хорошо, я сделал некоторые дополнения, а также..**

Экс 3-6: Lissajousfigures, script 2 (1/4)

Количество LINSPACE () шаги понижается до 40, чтобы лучше показать стрелки, которые используются ниже

Тело plot2d () сохраняется, напоминание будет сделано ручками

На рисунке ручка называется gcf(), после чего цвет цифры и фон может быть определен (дополнение к сценарию 1)

```
// handles_demo2-3.sce

// Two Lissajous figures, sin(t) & cos(3t) and   /
// sin(1.5t) & 0.5*cos(1.5t), with plot edited   /
// using handles                                 /

clear,clc,clf;

// Plot Lissajous figures:
//-------------------------------
x=linspace(0,6,40)';    // 40 steps to allow arrows
sines = [sin(x) sin(1.5*x)];              // First figure
cosines = [cos(3*x) 0.5*cos(1.5*x)];   // Second figure
plot2d(sines,cosines,rect=[-1.1,-1.1,1.1,1.1])

// Add background color:
//-------------------------------
f=gcf();                          // Get Figure handle
  f.background=color('grey'); // Grey background color
```

Экс 3-6: Lissajousfigures, script 2 (2/4)

Призываю ось справиться с GCA (), затем редактировать два фигуры Lissajou

p1 & p2 являются соединениями, дети к осям

Графики являются ломаными линиями и внуки к осям

Название и оси отметок должны быть сначала добавлены, после чего они могут быть отредактированы

Напомним, что Название является дочерним для осей

Проверить с Помощь / graphics_fontsfor сведения о шрифтах

```
// Edit Lissajous figures:
//-----------------------------
a=gca();                // Get Axes handle
p1=a.children;      // sin(1.5x), 0.5*cos(1.5x)
  p1.children(1).polyline_style=4;       // Arrow mode
  p1.children(1).foreground=2;  // Change color to blue
  p1.children(1).arrow_size_factor=2;  // Line thickness
p2=a.children;         // sin(x), cos(3x)
  p2.children(2).line_style=4;    // Dash-dot line
  p2.children(2).foreground=5;  // Change color to red
  p2.children(2).polyline_style=4;       // Arrow mode
  p2.children(2).arrow_size_factor=2;  // Line thickenss

// Add & edit title & labels:
//-------------------------------
xtitle('Two Lissajous figures', 'x-axis', 'y-axis');
  a.title.font_style=8;              // Font: Helvetica bold
  a.title.font_size=3;              // Increase title font size
  a.x_label.font_style=7;          // Font: Helvetica italic
  a.x_label.font_size=2;           // Increase x-label font
  a.y_label.font_style=7;          // Font: Helvetica italic
  a.y_label.font_size=2;           // Increase y-label font
```

Экс 3-6: Lissajousfigures, script 2 (3/4)

x- и y-ось отметки и знаки (легенды) добавляются

Оси этикетки, цвет шрифта и размер передопределены

Обратите внимание, что отметки и легенды (знаки) дети до осей, похожи на галочки

Легенда добавляется и редактируется.

```
// Edit ticks & marks (labels):
//-------------------------------------
a.x_ticks = tlist(['ticks','locations','labels'],...
   [-1.1,-.825,-.55,-.275,0,.275,.55,.827,1.1],...
   ['-1.1','-.825','-.55','-.275','0','.275','.55',...
   '.825','1.1']);
a.y_ticks = tlist(['ticks','locations','labels'],...
   [-1.1,-.825,-.55,-.275,0,.275,.55,.827,1.1],...
   ['-1.1','-.825','-.55','-.275','0','.275','.55',...
   '.825','1.1']);
a.labels_font_color=13;   // Change label color
a.labels_font_size=2;     // Increase label size

// Add & edit legend:
//-------------------------
legend(['sin(x), cos(3x)'; 'sin(1.5x), 0.5*cos(1.5x)'], 4);
leg=a.children(1);          // Get legend handle
   leg.font_style=2;        // Font: Times
   leg.font_size=2;         // Increase legend font size
   leg.font_color=1;        // Font color black
   leg.background=7;        // Yellow legend box fill
```

Экс 3-6: Lissajousfigures, script 2 (4/4)

Чтобы закончить, сетка включена "на" цвет линий редакции

Scilab не имеет эквивалента от Matlab "сетки на," это способ обойти проблему

```
// Add & edit grid:
//---------------------
set(gca(),'grid',[1 1]);          // Matlab's "grid on"
   a.grid(1)=color('green');   // Vertical line color
   a.grid(2)=color('green');   // Horizontal line color
```

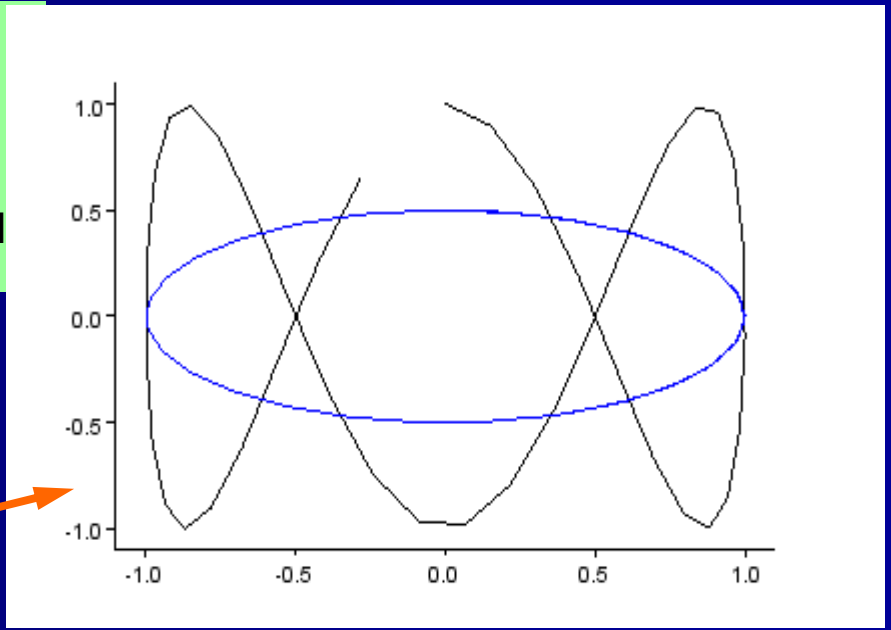
Были огромные проблемы, когда я впервые попытался включить ***gce()****, получить текущее лицо, команду в сценарии. Цвет фона не придумали, после Scilab был перезагружен, я не мог определить отметки и т.д.*
Извлеченные уроки : Убедитесь, что вы знаете, что вы делаете с GCE ()!
И последняя проверка на следующем слайде ...

Экс 3-6: Lissajousfigures, проверка

После всех этих изменений, давайте убедимся, что мы можем вспомнить основной plot,добавив следующие строки в конце сценария:

```
// Check default settings:
//-----------------------------
xdel();      // Delete Graphics Window
sda();       // Reset default Axes
plot2d(sines,cosines,rect=[-1.1,-1.1,1.1,1.1]
```

Когда мы запускаем script, Scilab кратко мигает модифицированным участком, * удаляет его, и помещает это окно вместо того. Основные фигуры Lissajou, кажется, в порядке *) Вы можете использовать функции пара drawlater () и drawnow (), чтобы избежать

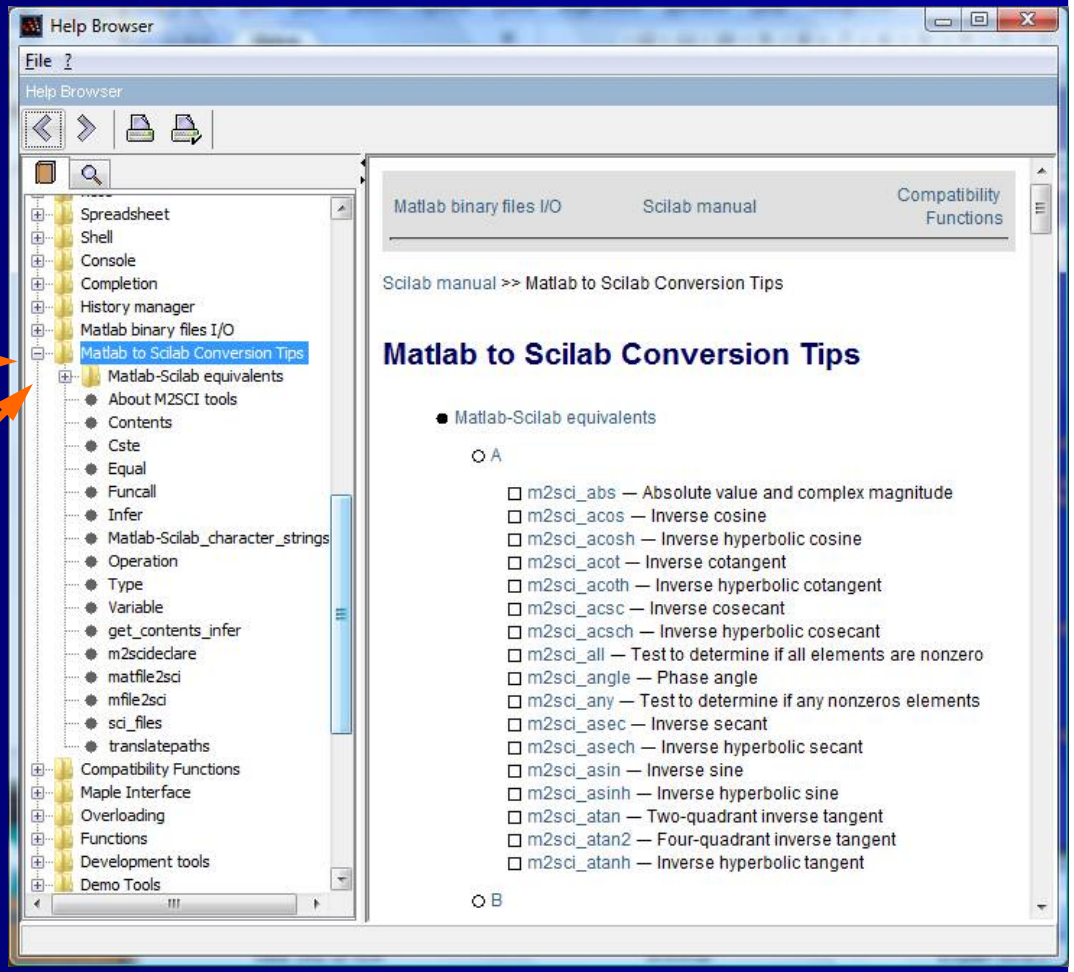


9. Преобразование Matlab файлы

- Встроенный переводчик Matlab-на-Scilab, кажется, работает и с вариантом ручного преобразования

Советы по Matlab в преобразования Scilab

Примерно на полпути вниз с помощью браузера вам надо найти Matlab в Scilab Советы преобразования. Вы увидите длинный список в m2sci_ ... типа функций
Нажмите на первую субпозицию, Matlab-Scilab эквивалентов, и вы получите список функций Matlab и их эквивалентов Scilab (отсутствующие файлы не включены, как текст () и Xlim () указано ниже)

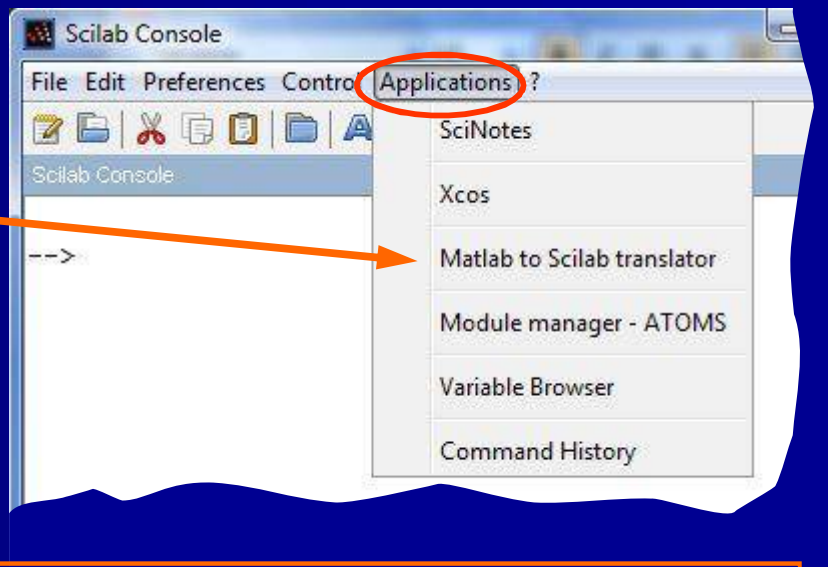


Использование встроенного Matlab-на-Scilabtranslator

Scilab может конвертировать Matlab годов. м в файлы. Мы не должны ожидать полностью успешный переход каждый раз. Thetranslated script, возможно, придется изменить вручную

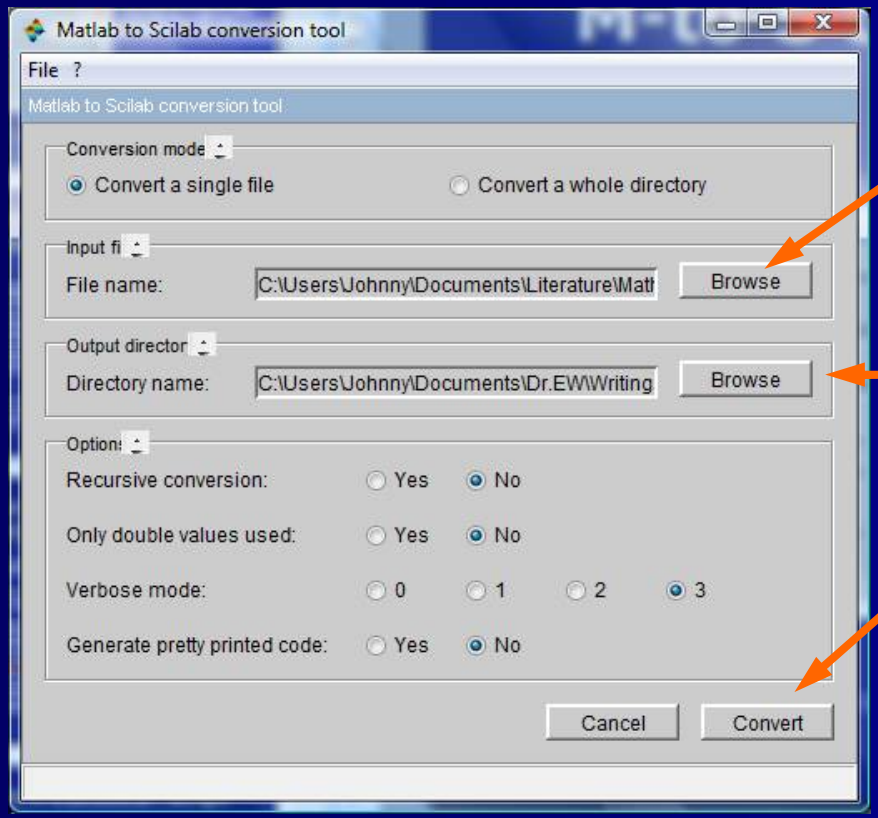
Мы начнем с открытия переводчика: *

На консоли выберите:
Приложения \ Matlab в Scilab переводчика
Что происходит дальше вы можете посмотреть на следующем слайде



*) Это раньше можно импортировать Matlab файлы непосредственно, но этот вариант сейчас не существует.

Переводчик M-к-S: процесс

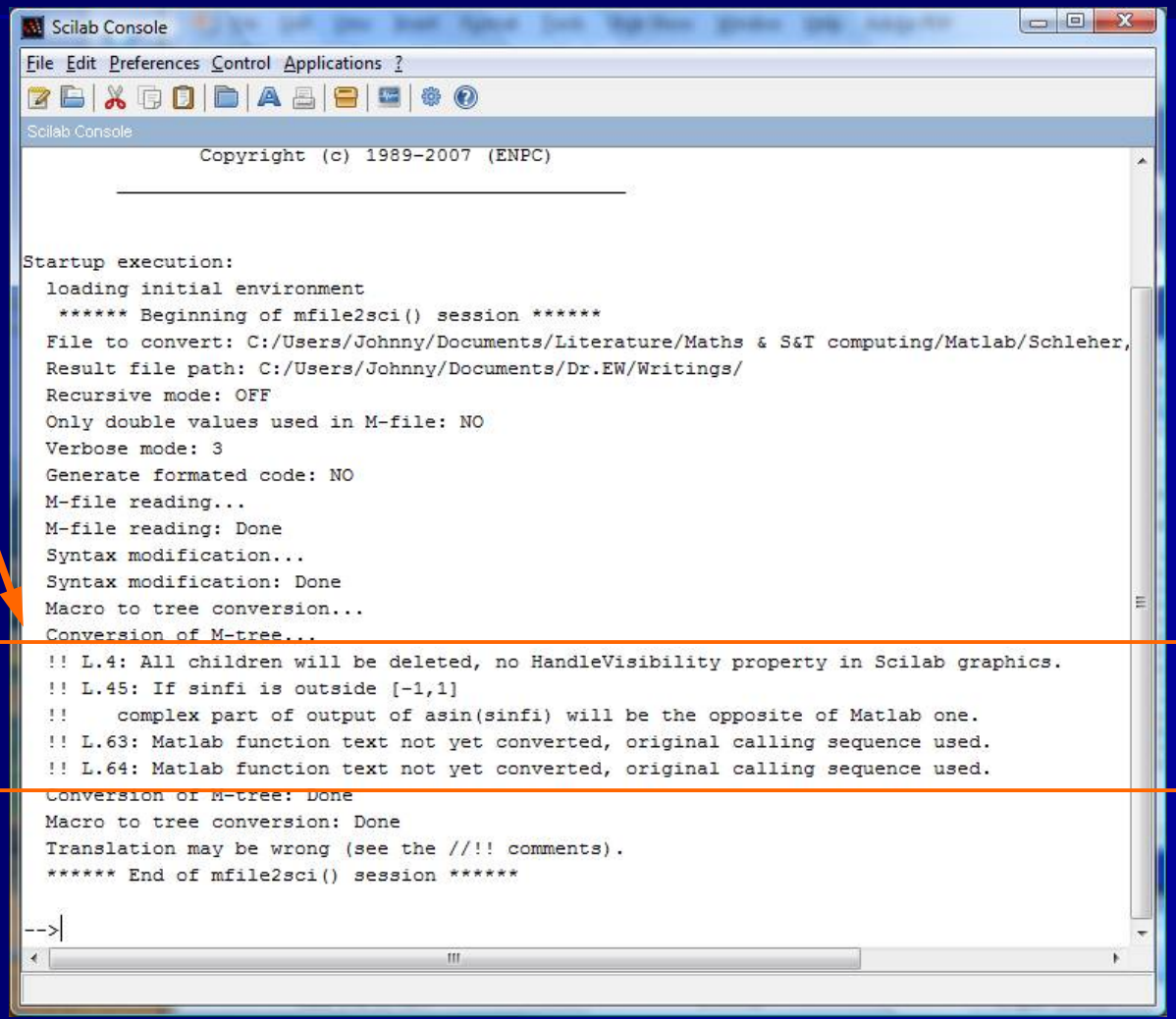


Откроется инструмент преобразования
. 1 Нажмите: (Имя файла) Обзор и определить файл, который вы хотите перевести
. 2 Нажмите: (имя каталога) Browseto указать, где поставить переведенный script (и связанные с ними продукты *)
. 3 Нажмите: Преобразование

*) Конверсию производит два текстовых документа и два сценария SCI

M-к-S переводчик: Сообщения на консоли

Scilab представляет список условий перевода, а также предупреждения о возможных ошибках на консоли. Предупреждения повторяются в виде комментариев в script (и на одном из текстовых документов) Затем открой те переведенный script в редакторе



Переводчик M-к-S: script (1/4)

Это script, которые представляет переводячик. Он содержит комментарии, которые могут или не могут иметь значение:

Предупреждение. Команда Matlab было ясно, CLC, КТМ;. Может иметь значение, если script редактируется ручками. В таком случае, попробуйте создать новый

```
// Display mode
mode(0);

// Display warning for floating point exception
ieee(1);

// Monopulse Antenna Pattern
// -----------------------------------

clear,clc,// !! L.4: All children will be deleted, no
HandleVisibility property in Scilab graphics.
clf;

// Normalized Aperture Width
na = 4;

// Sampling Frequeny=Number elements per norm
aperture
fs = 8;
```

Переводчик M-к-S: script (2/4)

Здесь все проходит гладко
Код, как ожидается,
представит суммы и разности
шаблонов для
моноимпульсной антенны
(отслеживание радар и т.д.)

```
// Norm aperture with N elements
N = fs*na;
xna = na*(-1/2:1/(N-1):1/2);

// Illumination Function

wxna(1,1:N/2) = ones(1,N/2);
wxna = mtlb_i(wxna,N/2+1:N,-ones(1,N/2));
wxnb(1,1:N/2) = ones(1,N/2);
wxnb = mtlb_i(wxnb,N/2+1:N,ones(1,N/2));

// Fill with M/2 zeros front and back

M = 1024;
xna = na*(-1/2:1/N+M-1:1/2);
wxna = [zeros(1,M/2),wxna,zeros(1,M/2)];
wxnb = [zeros(1,M/2),wxnb,zeros(1,M/2)];

// Beam Functions from -fs/2 to fs/2 in sine space

Nfft = max(size(wxna));
Esine = mtlb_fft(wxna,Nfft);
Esine = fftshift(Esine);
```

Переводчик M-к-S: script (3/4)

А вот несколько предупреждений. Может относиться к ошибкам округления

```
Esum = mtlb_fft(wxnb);
Esum = fftshift(Esum);

// Azimuth vector

sinfi = ((fs/4)*(-Nfft/2:Nfft/2-1))/Nfft;

// Azimuth vector in radians

// !! L.45: If sinfi is outside [-1,1]
// !!    complex part of output of asin(sinfi) will be the
opposite of Matlab one.
fi = asin(sinfi);

// Beam gain functions

Gfi = (Esine .*conj(Esine))/Nfft;
Gfs = (Esum .*conj(Esum))/Nfft;

Gfi = mtlb_i(Gfi,1:Nfft/2,sqrt(Gfi(1:Nfft/2)));
Gfi = mtlb_i(Gfi,Nfft/2+1:Nfft,-
sqrt(Gfi(Nfft/2+1:Nfft)));
Gfs = sqrt(Gfs);
```

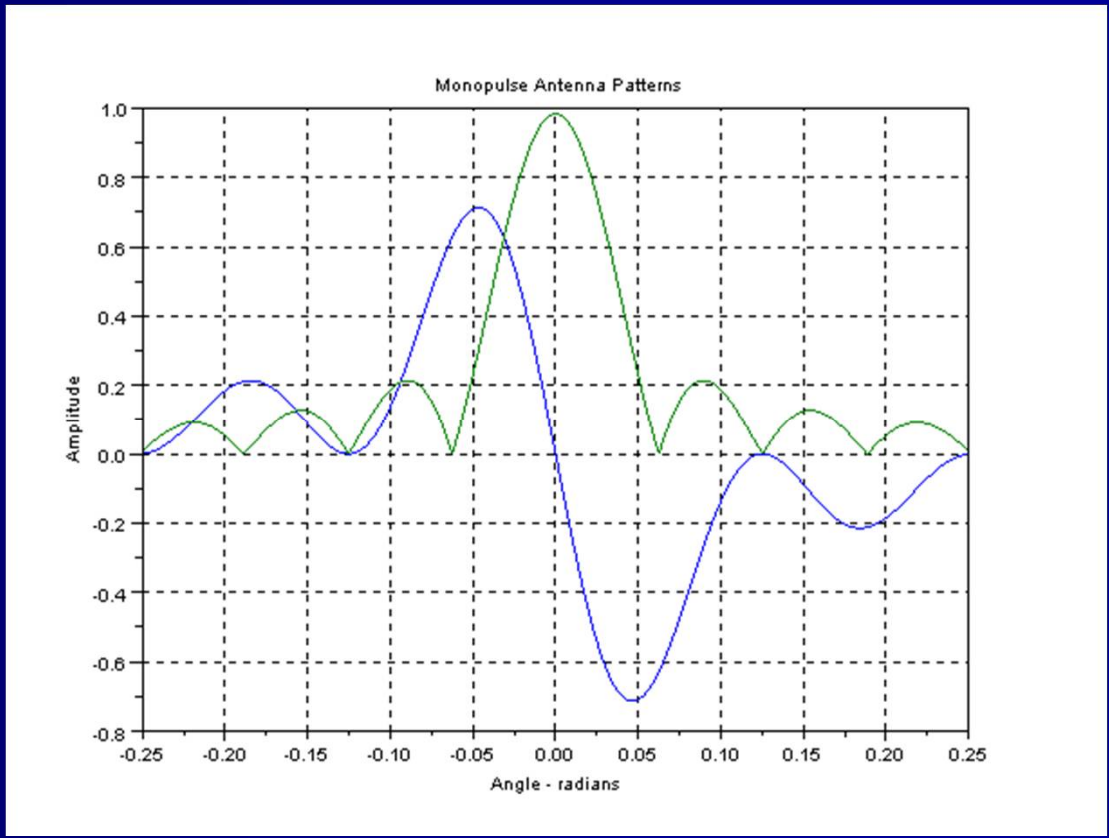
Переводчик M-к-S: script (4/4)

Здесь все проходит гладко
Код, как ожидается, представит суммы и разности шаблонов для моноимпульсной антенны (отслеживание радар и т.д.)

```
// Plot Monopulse Antenna Pattern

plot(fi,Gfi,fi,Gfs);mtlb_grid;
set(gca(),"data_bounds",matrix([-0.25,0.25,-0.8,1],2,-
1));
ylabel("Amplitude");
xlabel("Angle - radians");
title("Monopulse Antenna Patterns");
// !! L.63: Matlab function text not yet converted,
original calling sequence used.
text(0.04,0.8,"Sum Pattern");
// !! L.64: Matlab function text not yet converted,
original calling sequence used.
text(-0.22,0.6,"Difference Pattern");
```

Руководство преобразования (2/6): Случай № 1, plot

Были некоторые проблемы с этим преобразования:
-Распался длинный plot Matlab в **plot()**, но сокращенная форма не работал в Scilab
-Сначала я изменил ось Matlab в **axis()** , **rect()** с предыдущими местами. Теперь plot **plot()** работал
-Команды с метками вызвали проблемы. Причина была выше- я скопировал код Matlab и кавычки были наклеены неправильно



Руководство преобразования (3/6): Случай № 2, script и plot

Функция **pie()** не обсуждалась ранее, но ниже находится script Matlab, который рисует круговую диаграмму *

Функция **pie()** также существует в Scilab, разница в том, что в Scilab браузер не поддерживает pielabel Matlab-функцию ()

*) Более общий сектор графика будет представлен в примере 6-4

Руководство преобразования (2/6): Случай № 1, сценарий

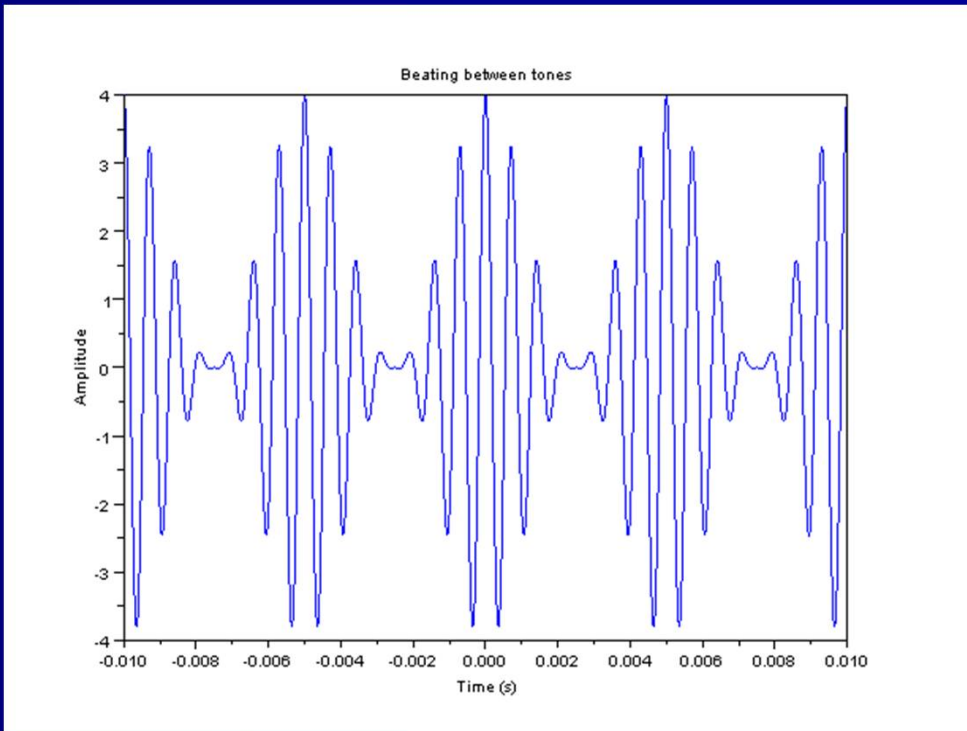
- Справа код Matlab (вверху) и его Scilab эквивалент (внизу). То, как я это сделал:
- Проверьте визуально разницы:
- Комментарии:% · / /
- Встроенные функции: пи ·% пи
- Команды земля, разделены на несколько строк. Отступы не нужны
- Изменил все, что я мог, то запустите скрипт и пусть отладчик Scilab в орать об остальном
- Часто Проверено с Help (особенно Matlab-Scilab эквивалента), чтобы понять сообщения об ошибках на консоли

```
% beating sinusoidal tones
%
t = linspace(-1e-2,1e-2,1001);
x = cos(2*pi*1500*t) +
cos(2*pi*1300*t);
m = 2*cos(2*pi*100*t);
plot( t, m, 'b:', t, -m, 'b:', t, x, 'k' ),...
   axis( [-0.01 0.01 -2.4 2.4] ),...
   title( 'Beating between tones' ),...
   xlabel( 'Time (s)' ),...
   ylabel( 'Amplitude' )
```

```
// M-to-S_1-modulation.sce

// beating sinusoidal tones   /

cear,clc,clf;
t = linspace(-1e-2,1e-2,1001);
x = cos(2*%pi*1500*t) +
cos(2*%pi*1300*t);
m = 2*cos(2*%pi*100*t);
plot( t, x.*m, rect = [-0.01 0.01 -2.4 2.4] )
   title( 'Beating between tones' )
   xlabel( 'Time (s)' )
   ylabel( 'Amplitude' )
```

Руководство преобразования (2/6): Случай № 1, график

- Были некоторые проблемы с этим преобразования:
- Я разделить команду Matlab задолго plot() но сокращенная форма не работал в Scilab
- Впервые я изменил ось Matlab axis() на rect()
- Команды подписей дали проблемы. Причина была свыше-что я скопировал код Matlab и кавычки были неправильными

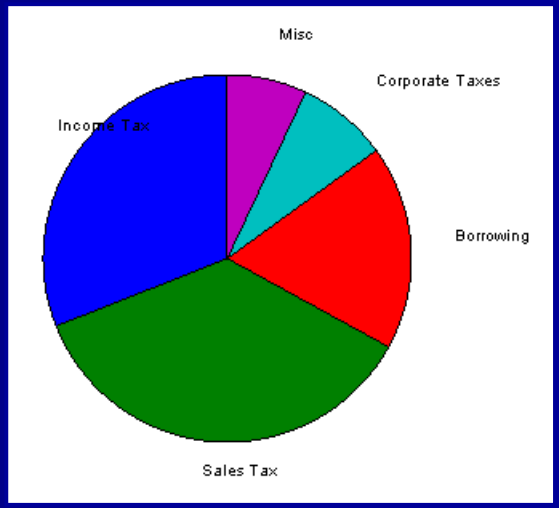


Руководство преобразования (3/6): Случай №2, сценарий, график

- **Функция** pie() **не обсуждалась ранее, но ниже находится в нескольких минутах сценарий Matlab, который рисует круговую диаграмму ***
- **Функция** pie() **также существует в Scilab, разница в том, что Scilab браузер не поддерживает функцию Matlab-** pielabel()

```
revenues = [31 36 18 8 7];
h = pie(revenues);
pielabel(h,{'Income Tax: ';'Sales Tax:
';'Borrowing: ';...
   'Corporate Taxes: ';'Misc: '});
```

```
// M-to-S_2-pie.sce

// Draw a pie graph with labels  /

clear,clc,clf;
revenues = [31 36 18 8 7];
pie(revenues,['Income Tax';'Sales
Tax';'Borrowing';...
   'Corporate Taxes';'Misc']);
```



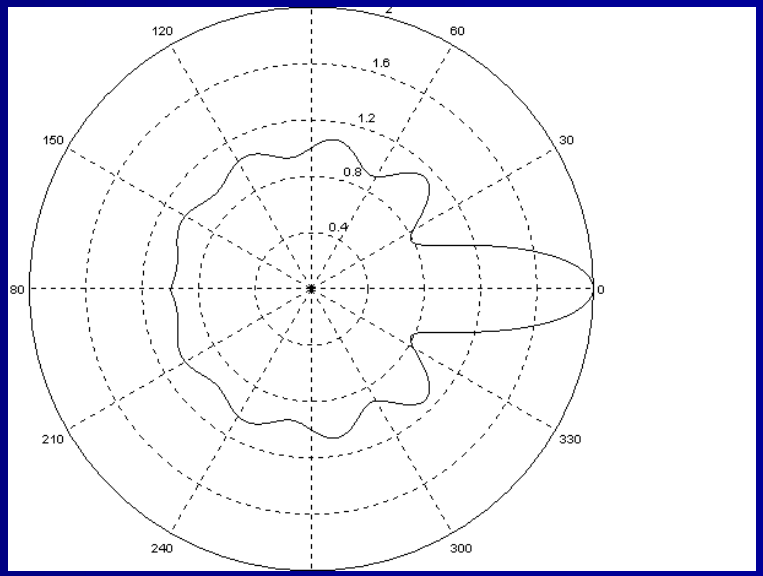
Руководство преобразования (4/6): Случай № 3, сценарий, графиу

В последнем случае, давайте посмотрим на смещенние функции sin

В этом случае проблема в том, что polardb () является неформальным созданием пользователями Matlab, который не поддерживают Scilab

```
x = -(5*2*pi):.1:(5*2*pi);
th = linspace(-pi,pi,length(x));
rho=((1+sin(x)./x));
polardb(th,rho)
```

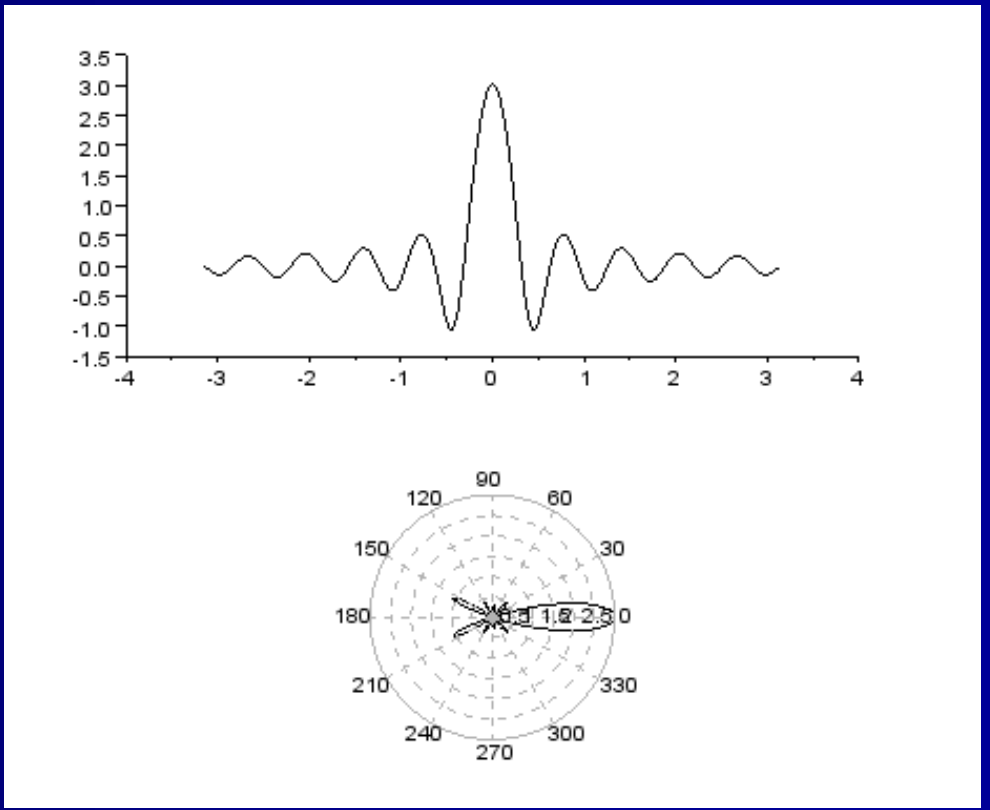
```
// M-to-S_3polarplot.sce

// Polar plot of 1+sin(x)/x   /

clear,clc,clf;
x = -(5*2*%pi):.1:(5*2*%pi);
th = linspace(-
%pi,%pi,length(x));
rho = 1+sin(x)./x;
polarplot(th,rho)
```



Руководство преобразования (5/6): Дело № 3, обсуждение

Полярный график с радиальными единицами в dB выглядит довольно "нелогичным", так как его боковые лепестки по всей видимости, указывает в неправильном направлении

```
// M-to-S_3polarplot.sce

// Polar plot of 1+sin(x)/x    /

clear,clc,clf;
x = -(5*2*%pi):.1:(5*2*%pi);
th = linspace(-
%pi,%pi,length(x));
rho = 10*log10((1+sin(x)./x));
subplot(211);
plot2d(th,rho)
subplot(212);
polarplot(th,rho)
```



Руководство преобразования (6/6): обсуждение и намеки

Руководство преобразование кодов Matlab для сценариев Scilab. Есть пользователи, которые утверждают, что делают это регулярно

Scilab для Matlab пользователей-учебники и форумы Scilab могут помочь в понимании различий

Некоторые функции Matlab просто не существует в Scilab (и наоборот).

Примерами являются axis(), compass(), feather(), fill(), nargin(), polar(), quad(), quiver(), stem(), stairs(), и waterfall()

Иногда альтернативные команды Scilab существует (например, plot2d2 Scilab() могут компенсировать лестницей MATLAB в ()), иногда нет. Если нет, то script должен быть переписан

Пользовательские функции Scilab должны загружаться с GETF (), в то время как Matlab не имеет отдельную функцию загрузки

Выполнения MATLAB в data.mshould могут быть обменены на Exec ('data.sci') в Scilab

Еще один случай из ручного преобразования будет представлен в примере 6-5 (Глава 19)

10. Подпрограммы

- Это обсуждение подпрограмм является прелюдией к управлению, которое будет обсуждаться в глава 11

Терминология

В главе 1, что Scilab не признает термин "подпрограммы", который относится к группе разнообразных конструкций, которые Scilab называет "функция". Более точно, речь идет о пользовательских функциях(UDF), выражениях, которые Scilab также делает, не знаю как.
Независимо от официальной терминологии Scilab, я- когда это возможно- использовал традиционный термин подпрограммы, так как это элегантный способ указывать на конкретные лица в компьютерных программах

Введение

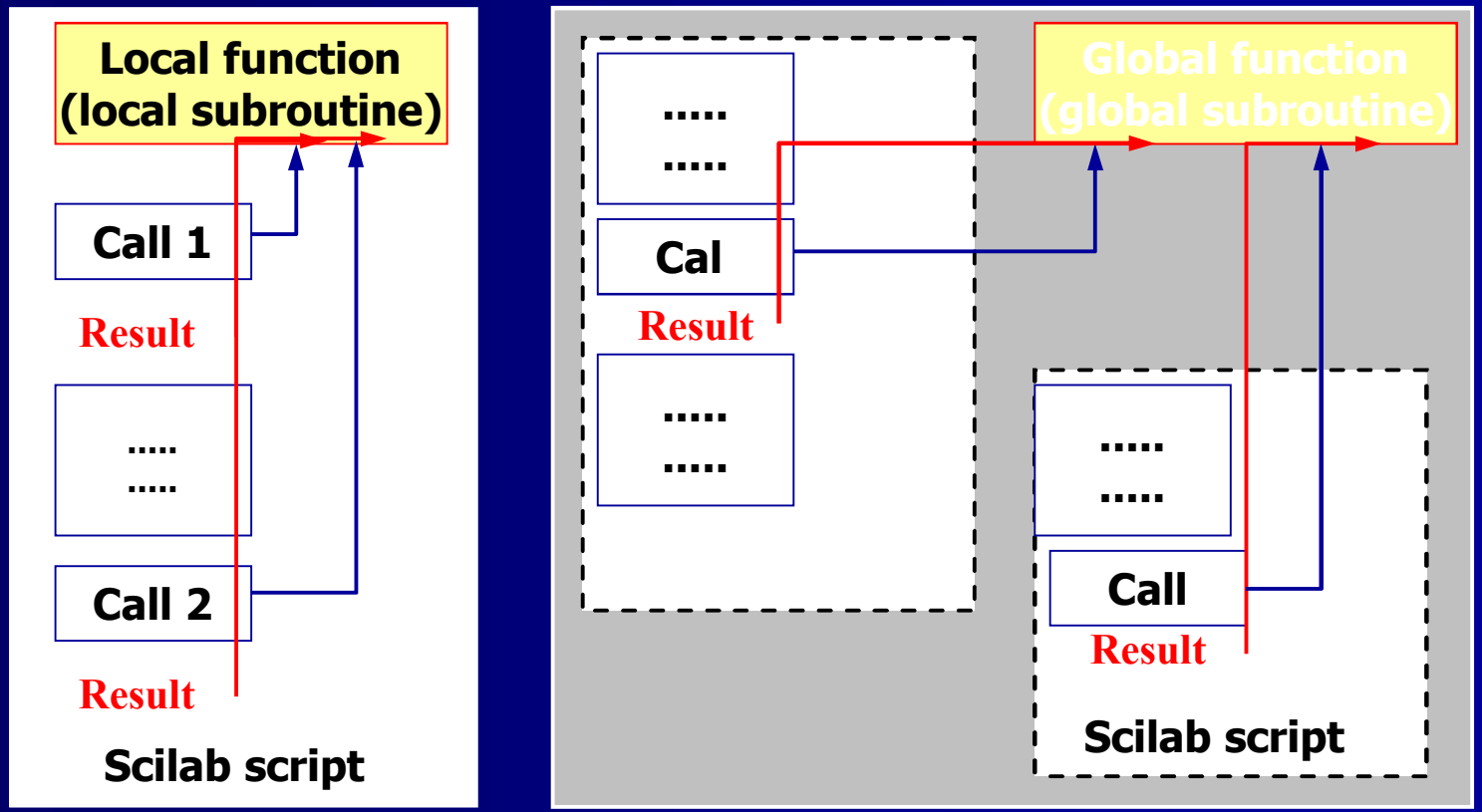
Напомним, что в Пример 1-3 , ввели понятие пользовательских функций (UDF)
Задача: Написать функцию, которая вычисляет площадь треугольника с известными длинами сторон
Функция вводится на редакторе
Затем загружается в консоли с помощью команды редактора
Выполнить / ...file с эхом
Функция выполняется, введя имя функции и входные параметры (длины сторон) на консоли

```
function A = triangle_area(a,b,c)

// The function 'triangle_area' calculates the   /
// area of a triangle with side lengths a, b, c.  /

funcprot(0)
p = (a+b+c)/2  // p = half perimeter
A = sqrt(p*(p-a)*(p-b)*(p-c))
endfunction
```

```
 -->function A = triangle_area(a,b,c)
-->// The function 'triangle_area' calculates the   /
-->// area of a triangle with side lengths a, b, c.  /
-->funcprot(0)
-->p = (a+b+c)/2  // p = half perimeter
-->A = sqrt(p*(p-a)*(p-b)*(p-c))
-->endfunction

-->triangle_area(4,5,6)
 ans  =

 9.9215674
```

Локальные и глобальные функции (подпрогр аммы)

Локальные функции встроены в script и действует в одиночку, глобальные функции сохраняются отдельно и доступны для любого сценария



Локальные и глобальные переменные

Вы столкнетесь с условиями локальных и глобальных переменных, и они должны дать краткие разъяснения:

-Как и функций, Scilab имеет два типа функциональных переменных , локальные и глобальные:

Локальные переменные ограничены конкретной функции

Глобальные переменные доступны, и могут быть изменены всеми функциями, в которых переменная была объявлена глобальной

-Передача параметров с помощью командного окна (консоли) переменных и глобальных переменных не слишком очевидно. Глобальные переменные, в частности, могут привести к ошибкам, которые трудно обнаружить

-По указанным причинам использование глобальных переменных должно быть ограничено до минимума

В заключение: мы будем рассматривать только локальные переменные , которые по умолчанию в Scilab.

Подпрограммы, более формально

В общем, когда подпрограмма имеет несколько входных аргументов (in_arg1, in_arg2, ...) и возвращает несколько аргументов вывода (out_arg1, out_arg2, ...), структура:

```
Функция [out_arg1, out_arg2, ...] = ...
funktion_name (in_arg1, in_arg2, in_arg3, ...)
out_arg1 = выражение для первого выходного аргумента;
out_arg2 = выражение для второго выходного аргумента;
...
EndFunction
```

Структурой границы является функция .... EndFunction ограничители
Входные параметры сгруппированы в скобках (скобки), выходных параметрах в квадратных скобках (не требуется для одного выходного параметра)
В обоих случаях аргументы разделяются запятыми

На выходных аргументов

Пример справа подчеркивает основной способ, в котором Scilab управляет выходные аргументы подпрограмм.
Когда вам нужно влиять на управление входными и выходными переменными, Scilab предлагает функции **argn ()**, varargin (), и varargout ()

```
-->function [y1,y2] =
myfunc(x1,x2);
-->y1 = 3*x1;
-->y2 = 5*x2+2;
-->endfunction

-->myfunc(4,7)
 ans  =
            3x4=12
    12.

-->y2 = myfunc(4,7)
 y2  =
            3x4=12
    12.

-->[y1,y2] = myfunc(4,7)
 y2  =
            5x7+2=37
    37.

 y1  =
            3x4=12
    12.
```

With no output argument defined, the first output argument is returned in the ans variable

The same answer is returned when only one output argument is defined

With both output arguments defined, the result of the computation is returned in full

Vector аргументы

Эта функция * использует векторы в качестве аргументов входных и выходных
Функции впервые определены
После этого выходные аргументы-операции, которые вы-определяете
Следующие входные аргументы (строка векторы) вводятся
В конце выполнения функции
Функция в линию Deff () является специализированной формой местных функций

```
-->// Define local subroutine cross_product
-->function [x] = cross_product(a,b)
-->    x(1) = (a(2)*b(3) - a(3)*b(2))
-->    x(2) = (a(3)*b(1) - a(1)*b(3))
-->    x(3) = (a(1)*b(2) - a(2)*b(1))
-->endfunction

-->// Plug in numeric values
-->a = [-2 5 8];
-->b = [7 -13 -5];

-->// Executing the subroutine
-->c_prod = cross_product(a,b)
 c_prod  =

   79.
   46.
  - 9.
```

x(1)=5*(-5)-8*(-13)
x(2)=8*7-(-2)*(-5)
x(3)=-2*(-13)-5*7

*) Здесь я использую термин «функция», так как код является независимым и не вызывается в главной программе.

Demo (1/2): script

Задача: Вычислить и построить параболу, найти положительный корень

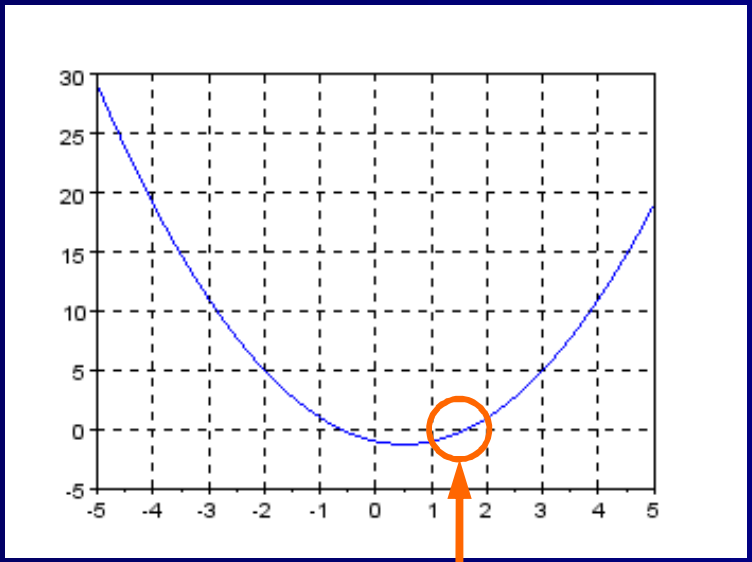
Вот подпрограмма открывается в первый раз с помощью ввода аргумент x

Вот подпрограмма открывается еще дважды, сначала с входным аргумента a, потом

Интересный способ найти корневой каталог. Позже мы будем делать то же с помощью fsolv ()

```
// subroutine1.sce

// Compute & plot the function y = x^2-x+1 in the  /
// range [-5,5] and determine its positive root.   /
// Assume we know the root lies in the range [1,2]  /

clear, clc, clf;

// SUBROUTINE para():
//-------------------------------
function y=para(x);     // Subroutine declaration
y = x^2-x-1             // Equation (parabola))
endfunction

// MAIN script:
//-------------------
x = linspace(-5,5,100); // Range of interest
plot(x,para)            // Call subroutine and plot
xgrid;                  // Add grid

a=1; b=2;               // Search limits
while b-a > 10^(-4)     // Accuracy for searching root
  c = (a+b)/2;          // Midpoint of limits
  if para(a)*para(c)>0 then   // IF (lower)*(midpoint)
                              //    is positive
    a=c;                // THEN change lower limit
  else
    b=c;                // ELSE change upper limit
  end
end
disp("The root lies between "...   // Output root limits
  +string(a)+" and "+string(b))
```

Demo (2/2): plot, распечатка и комментарии



The root lies between 1.617981 and 1.618042

Это demo было заимствовано из брошюры **"Scilab pour les Lycées"**

Обратите внимание, что вызов команды был сокращен в максимально возможной степени. Вместо того, чтобы мы могли бы написать plot (x, para):

Подпрограммы должны быть объявлены до вызывающего основной части сценария

Позже мы увидим, сценарии с несколькими подпрограммами, отличаются друг от друга по именам. Подпрограммы также могут быть вложенными (следующий слайд) и могут вызывать другие подпрограммы

Deff () примитив

Deff () примитив может быть использован для определения простые функции (она напоминает инлайн Matlab-функцию ()) Deff (), следовательно, используется в местных подпрограммах
Синтаксис: Deff ('y = имя_функции (x1, x2, ...)', 'y = функция выражение')
Ниже то же самое уравнение вычисляется на консоли с функцией и Deff () альтернативой:
Обратите внимание на точку с запятой! (Scilab понимает, если вы пропустите его после второго y выражения)

```
-->function y = nested(x)
-->    a = sin(x) + 2*%pi;
-->        function y =
inner(x);
-->            y = x^2 -
sqrt(x);
-->        endfunction
-->    y = inner(a) + 3^2;
-->endfunction

-->value = nested(%pi/3)
 value  =

   57.437413
```

```
--> (sin(%pi/3)+ 2*%pi)^2 - sqrt(sin(%pi/3) + 2*%pi)
+ 3^2
 ans  =

    57.437413
```

Примитив deff()

- The `deff()` примитив , но может быть использована для определения простые функции (похоже на функцию Matlab's `inline()`)
- `deff()` следовательно, используется в местных подпрограмм
- Синтаксис

  `deff ('y = function_name(x1,x2,...)', 'y=function expression')`

- Ниже то же самое уравнение вычисляется на консоли с функцией и Deff () альтернатив: `deff()` :

```
-->deff('y = f(x)','y = x^2+x-1')

-->f(2), f(-5)

ans  =
   5.
ans  =

   19.
```

```
-->function y = g(x); y = x^2+x-1 endfunction

-->g(2), g(-5)
ans  =

   5.
ans  =
   19.
```

Обратите внимание на точку с запятой! (Scilab понимает, если вы пропустите его после второго y выражения)

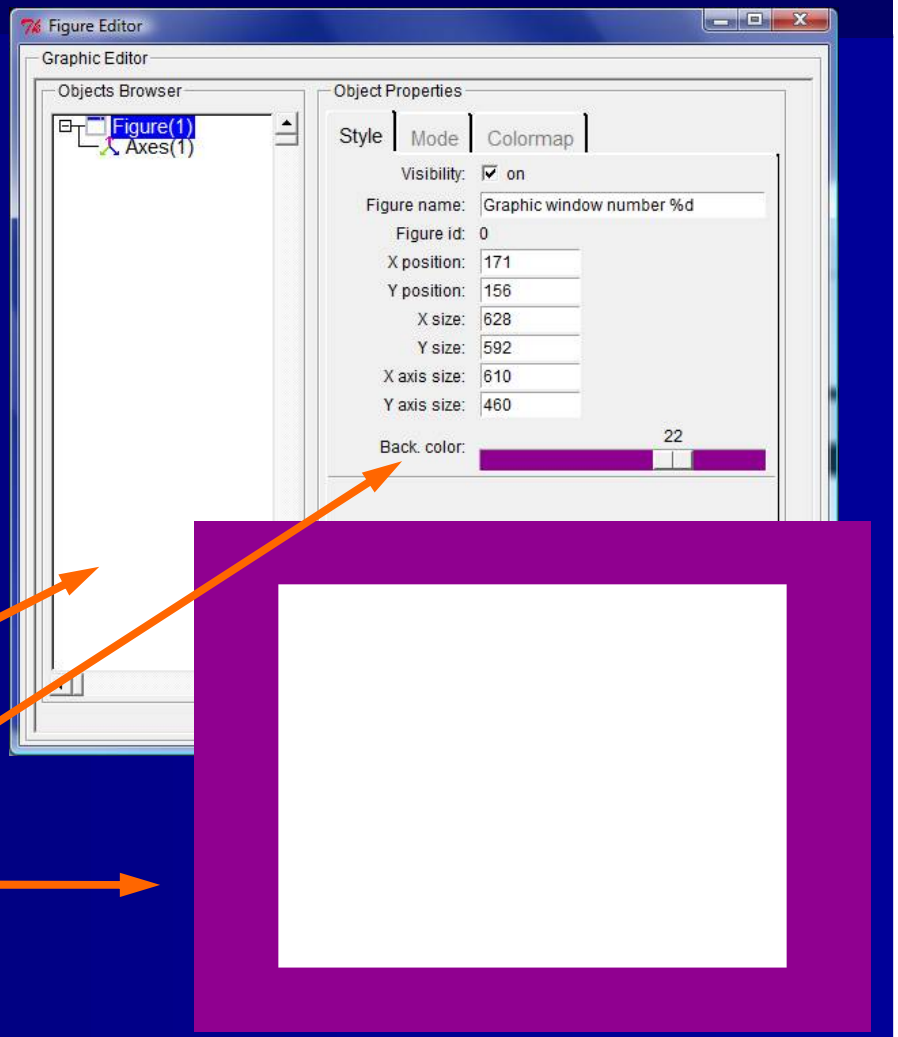
Глобальные подпрограммы: window demo (1/4)

Это demo создает глобальную подпрограмму (функцию) для многоразового графического окна

Во-первых, window графика открыто Scilab (одной альтернативой является команда gcf(); на консоли). Пустая Графика.Window всплывает.

Далее, в окне с графикой, надо наажать: Редактировать / Рисунок propertiesto открыть редактор рисунка (это было объяснено выше)

В-третьих, выбрать подходящий цвет. Назад (например 22 для церковной фиолетовый), и вы можете увидеть кадр в окне с графикой (бар идет только до 32, серый это не вариант)



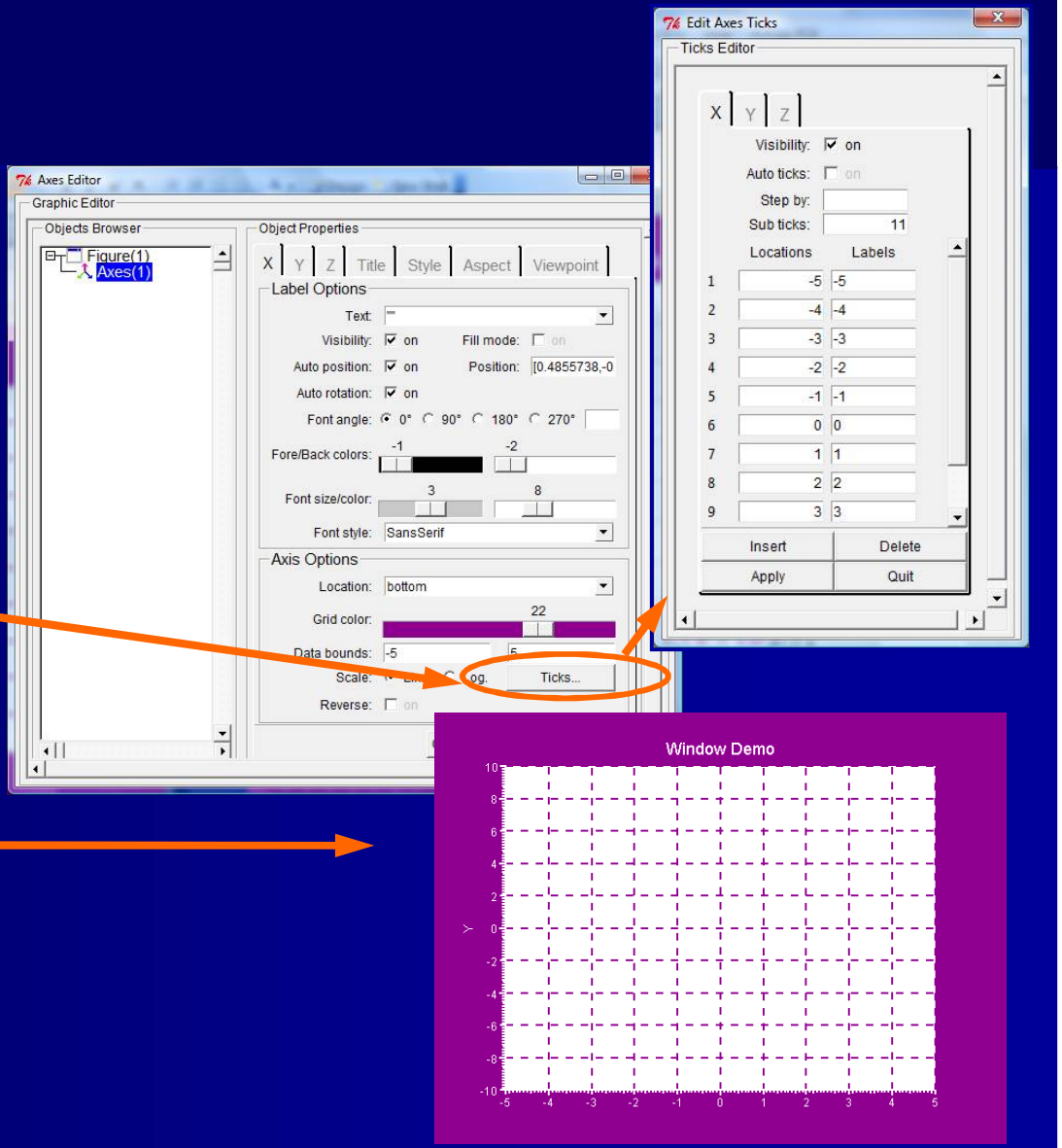
Глобальные подпрограммы: window demo (2/4)

В-четвертых, нужно играть с рис редактором, довольно долгое время, чтобы заполнить все детали

Обратите внимание на галочку...кнопку на графическом редакторе. Открывается отдельное window, в котором можно определить этикетки линий сетки

Позже, когда все детали совпадают, вы достигнете что-то вроде этого

Наконец, сохраните его с помощью окна с графикой. Нажмите: File /



Глобальные подпрограммы: window demo (3/4)

Scilab отвечает на консоли:

```
Figure saved.
```

Так что в [illegible]граммы, я назвал его window_demo.scg. Обратите внимание на окончание. **Не .sce или .sci**. g -для графики

Нам нужен главный script, который использует глобальную подпрограмму

Обратите внимание, что я называю ручку e = GCE Entity (). Это упрощает сравнение с необходимым путем, если называть ось обработки, как это было сделано в главе 7

```
// reuse_function.sce

// Reusing graphics function,            /
// defenitions with handle commands  /

clear,clc;

// SUBROUTINE, load function:
//---------------------------------------
load('window_demo.scg');

// MAIN, define and plot:
//-------------------------------
x = [-4:0.01:4];              // Horizontal extension
y1 = 2*sin(x) - 2;            // First equation
plot2d(x,y1, style=2);        // First plot, blue
  e=gce();                    // Get Entity handle
  e.children.thickness=5;     // Polyline size
y2 = 2*cos(2*x) + 6;          // Second equation
plot2d(x,y2, style=3);        // Second plot, green
  e=gce();                    // Get Entity handle
  e.children.thickness=5;     // Polyline size
```

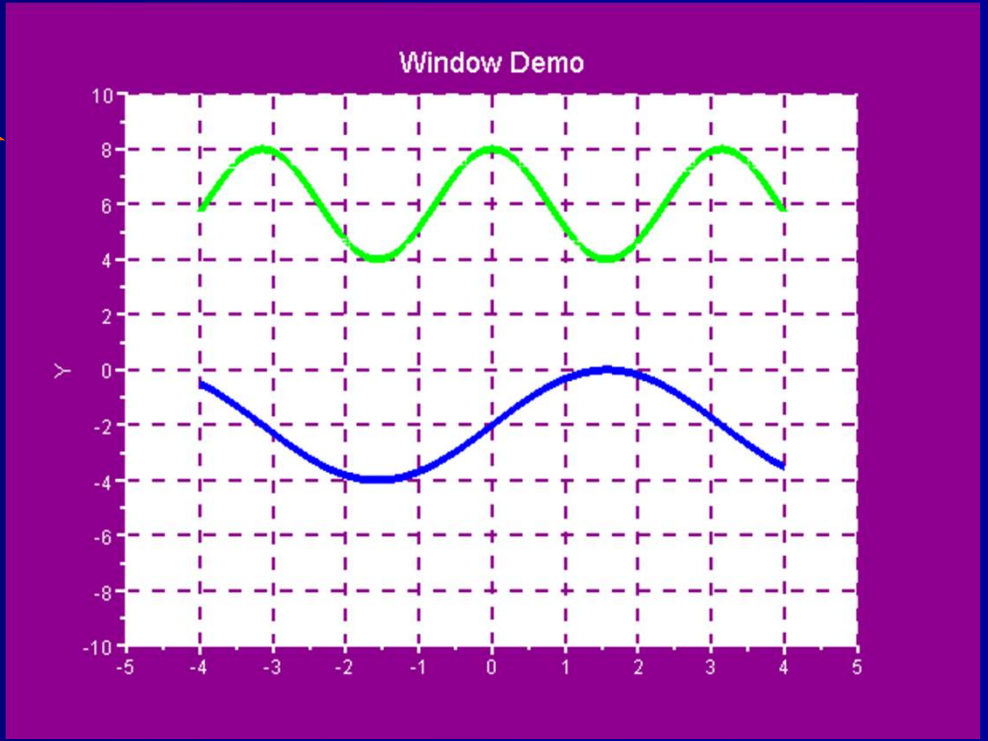
Глобальные подпрограммы: window demo (4/4), plot

У вас есть проблемы на участке?

Если да, то убедитесь, что вы определили window правильно

Например, если вы не правильно заполнить обе стороны в списке Редактировать Оси,поставив галочки там,где будут какие-то смешные места сетки

Также уверен, что границы данных ([-5,5], [-10,10]) определяются в окне Оси Editor



Я набрал неправильный GCA ручкой вызова () и Scilab разбился. Перезагрузка ...

Комментарий: несколько участков с помощью одной команды

Можно построить несколько графиков с помощью одной команды, определяя аргументы функции в качестве векторов столбцов

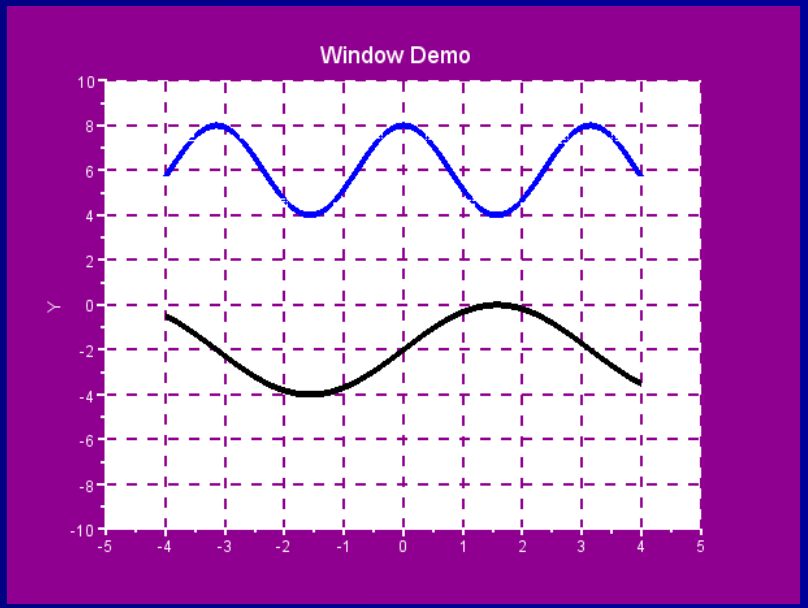
Вот измененный plot команды, в предыдущем окне demo сжимается до

plot2d (x, [y1 ', y2'])

Результат показан здесь. Scilab автоматически выбирает различные цвета для графиков

Scilab предупреждает, если графическое window пристыковывается.

```
y1 = 2*sin(x) - 2;            // First equation
y2 = 2*cos(2*x) + 6;          // Second equation
  plot2d(x,[y1',y2'])         // Plot both
  e=gce();                    // Get Entity handle
  e.children.thickness=5;     // Polyline size
```



11. Управление потоком

- **Управление потоком (условные ветвления, программирования) принесли важные новые структуры**

Введение

Теперь мы подошли к линии, которая отделяет мальчиков от мужчин

Некоторые примеры использования управления потоком (также известные как условное ветвление и программирование) придумали в предыдущих главах, но природа условных команд отраслевых, как если ..., то ... еще не обсуждалась *

Управление потоком вместе с подпрограммами необходимо для серьезного практического моделирования

Поэтому мы будем вырабатывать правильный (хороший) взгляд на наиболее важные аспекты управления потоком

Однако следует отметить, что операции цикла медленные. Мы должны стремиться к векторизованности операций, если задача требует много итераций цикла (есть краткое обсуждение по этому вопросу в главе 18)

*) Историческая справка: Конрад Цузе, немец, которые построил первый настоящий компьютер во время Второй мировой войны (с помощью более 2000 реле, потому что он не доверял вакуумным трубкам).

Поток управляющих конструкций

Ниже приведены основные конструкции, с которые вы должны быть знакомы:

Команды :

for … (if … else) … end
while … (if/then/else) … end
if … (elseif … else) … end
select … case (… else) … end
break … continue
try … catch … end

Операторы сравнения::

==	equal to
<	smaller than
>	greater than
<=	smaller or equal to
>=	greater or equal to
<> or ~=	not equal to

Логические операторы:

&	and
| or /	or
~	not

Логические / Логические константы:

%t or %T	true
%f or %F	false

for … end

for …end цикл повторяет группу операторов заданное число раз. Общее выражение для … конце концов, это:

для переменной = initial_value: шаг: FINAL_VALUE для
/ / Foo
конец

Нет запятой!

Как видно здесь:

```
// for-end_demo.sce

// Compute the square root and
square  /
// of odd integers from 1 to 8
/

n = 8;
for k = 1:2:n
   root = sqrt(k);
   quadrat = k^2;
   disp([k, root, quadrat])
end
```

Квадрат" не может использоваться в качестве имени переменной с площади () является функцией Scilab

```
 1.   1.   1.

 3.   1.7320508   9.

 5.   2.236068   25.

 7.   2.6457513   49.
```

for ... if ... else ... end

for ... end **могут быть вложены с** ***If / Else*** **условия для обеспечения исполнения альтернативных утверждений:**

```
альтернативных утверждений :
для переменной  = initial_value: шаг:
FINAL_VALUE для
if условие, если
/ / Foo
еще
/ / Foo
конец
конец
```

Следующие несколько слайдов prodemонстрируют случаи, когда случайный гауссовский "шум" генерируется, отсортировывается и сообщается устной и с графиком

for … if … else … end: demo, script (1/3)

Только переменные, нечего комментировать

```
// for-if-else.sce

// ------------------------------------------------------------------
---- /
// The script generates Gaussian noise around a fixed signal.
/
// Each sample ("signal") is sorted according to whether it
/
// is within, above or below default variance limits (+/-1). The
/
// result is reported verbally with strings and is also plotted
/
// ------------------------------------------------------------------
---- /

clear,clc,clf;

// Define variables:
// ------------------------
n = 500;        // # of for…end loops
above = 0;      // Signals above upper variance limit
below = 0;      // Signals below lower variance limit
within = 0;     // Signals within variance limits
ave = 3;        // Mean (average)
x = [];         // x axis vector
```

for ... if ... else ... end: demo, script (2/3)

Случайное генерирование , как обсуждалось раньше

Обратите внимание, как сигнал матрицы считывается с элемент- на-элемента как J идет от 1 до n

```
// Generate signal:
// ---------------------
dt = getdate();                          // Get date
rand('seed',(531+n)*dt(9)+dt(10));   // Initialize random
generator
signal= ave + rand(1,n,'normal');      // Shifted Gaussian signal


// Sort signal:
// ----------------
for j = 1:1:n
  if signal(1,j) > ave+1 then      // Default variance = +/-1
      above = above + 1;          // Count signal > mean+var
    elseif signal(1,j) < ave-1      // Default variance = +/-1
      below = below +1;           // Count signal < mean-var
    else                          // If within variance limits
      within = within + 1;         // mean-var <= signal <=
mean+var
  end
end
```

for ... if ... else ... end: demo, script (3/3)

Дисплей на консоли: общий, средний, и пределы дисперсии. Это особая форма нескольких участков обсуждалась ранее и ее стоит иметь в виду

```
// Display result:
// -------------------
disp(['Result from generating', string(n), 'Gaussian distributed
samples'])
disp(['(signals) with mean =' string(ave) 'and variance = 1:'])
disp([' -' string(within) ' samples were inside variance limits,'])
disp([' -' string(above) 'above upper variance limit, and'])
disp([' -' string(below) 'below lower limit'])

// Plot result:
// ---------------
x = [1:1:n];                          // Array for x axis
y1 = ave*ones(1,n);                   // Array for mean value
y2 = (ave+1)*ones(1,n);               // Array for upper variance limit
y3 = (ave-1)*ones(1,n);               // Array for lower variance limit
rect = [0,ave-4,n+1,ave+4];           // Set pot window
plot2d(x,signal,2,"011"," ",rect)     // Plot samples
plot2d(x,y1,5,"000")                  // Plot mean value
plot2d(x,y2,3,"000")                  // Plot upper variance limit
plot2d(x,y3,3,"000")                  // Plot upper variance limit
legend('signal','average','variance');
xtitle('GAUSSIAN RANDOM SAMPLES','Sample #','Sample value')
```

for ... if ... else ... end: demo, print & plot

!Result from generating 500 Gaussian distributed samples !

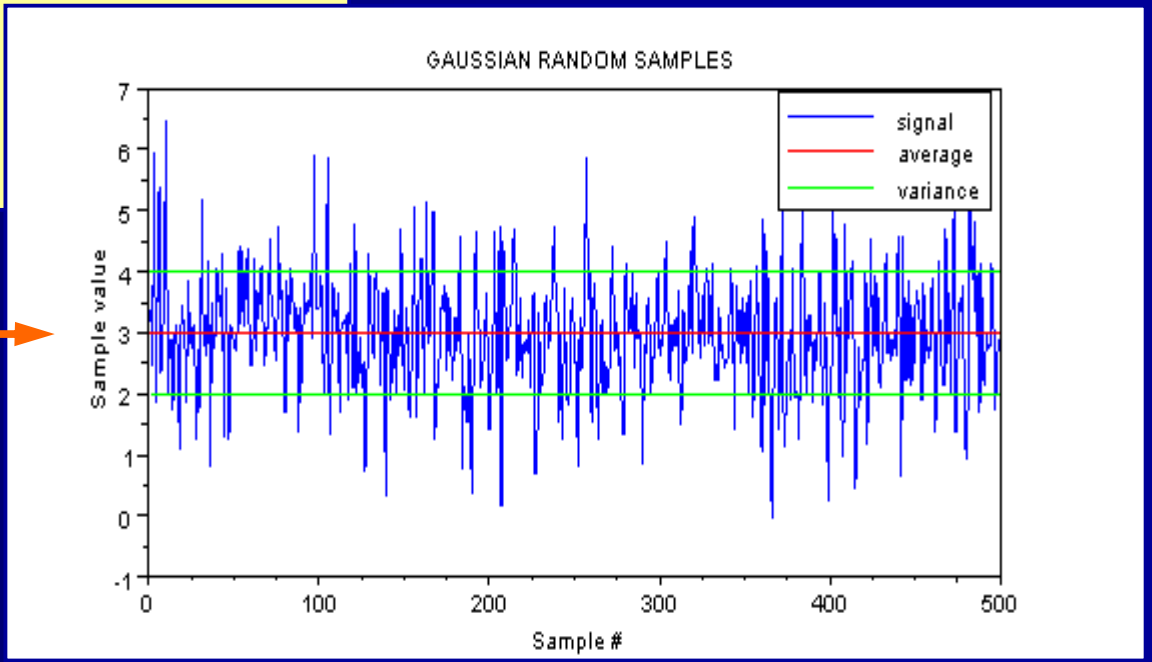
!(signals) with mean = 3 and variance = 1: !

! - 348 samples were inside variance limits, !

! - 75 above upper variance limit, and !

! - 77 below lower limit !

69.6% образцов находятся в пределах границ ± 1σ. Вполне хорошо для 500 образцов



Можно видеть, что существует один или два образца вне лимита ± 3σ, как и должно быть

for … if … else … end: comments

Обратите внимание, как случайные данные, собранные с сигнала **(:,n)матрицы**. Сравните это с x = [x, k] постройте то, что в дальнейшем будет используется в связи с обсуждением разрыва и продолжения.

Этот пример был опытом, сводящим с ума: я просто не мог понять, как сделать график. После нескольких часов попыток я нашел проблему: я положил случайную функцию внутри для … ENDLOOP

Это держало меня на ложном пути слишком долго. Правильно собранные данные были разрушены, когда он был доставлен для … конца цикла. Тем не менее, этого не произошло, если бы была команда отображения внутри цикла. Например, нет запятой после сигнала (:, п). Говорите совпадение …..

Проблема решилась, когда я, наконец, понял, что надо отделить генерации данных от данных сортировки петли

Извлеченные уроки: Будьте осторожны с тем, что вы помещаете внутрь цикла

while … end

В то время как ... конец цикл повторяется неограниченное число раз под управлением логического условия

Общая форма время ... endis:

в то время как/ / Foo

/ / Счетчик цикла, то есть
количество = рассчитывать 1;

Код вправо определяет, из какой стоимости к выра

2 ^ (-к) <=% EPS

Будьте осторожны с состоянием, его можно easilylock модел
бесконечном цикле

```
-->k = 1;

 -->while 2^(-k) >
%eps
-->k = k+1;
-->end

-->k
 k  =

   52.
```

while … if /then/else … end

While … end условие может быть вложено с дополнительными if … then … elseinstruction:

```
в то время как Condition_1 a
если condition_2 затем, если то
/ / Foo
еще
/ / Foo
конец
/ / Foo
конец
```

Функция на следующем слайде для игры, в которой пользователь должен угадать случайное число, которое компьютер рисует. Игра заканчивается только с правильным предположением

while ... if /then/else ... end: demo

```
// game.sci

// The function draws a random number in the    /
// range [1,M] that the user should guess.      /
// Game finishes when correct number is found   /

clear,clc;

M=30;                              // Upper limit of numbers
number=floor(1+M*rand());      // Draw a random
number
disp('Guess a positive integer in the range ');
disp([1,M]);                 // State range of random numbers
guess=input('You guess: ');       // User's guess

while (guess~=number)          // Start while condition
  if guess>number then         // Start if-then-else
    disp('Number is too big');
  else
    disp('Number is too small');
  end                                // End if-then-else
  guess=input('You guess: ');    // User's next guess
end                                  // End while condition
disp('Correct!');
```

Loop

Сохраните script, загрузите его в Scilab (на редактора), введите имя функции на консоли

```
-->guess_a_number

 Guess an integer in the range

      1.    30.
You guess: 15

 Number is too small
You guess: 22

 Number is too big
You guess: 17

 Number is too small
You guess: 19

 Correct!
```

Комментарии на интерактивности

)

В предыдущем demo показали примеры интерактивного использования строк

- Поручить пользователю: **disp(‘Guessan integer’)**

- Чтобы принять ввод пользователя: **guess = input(‘Youguess: ’)**

Для пользователя **input()** быстро, но не очень ясно, так как текстовая строка появляется только до -они должны, по крайней мере, мигать. Поэтому следует попытаться найти выразительные текстовых сообщений. Возможно, следующий будет лучше.

guess = input(‘Now, Sir/Madame, type your guess: ’)

Интерактивные текстовые строки являются простой формой человеческо-машинных интерфейсов; Графические интерфейсы пользователя (GUI) являются более передовыми и будут упомянуты в главе 15 (был случай, уже в Ex 1-3)

foo ... do ... end

Ключевое слово можно использовать внутри для отделения определения переменной цикла (состояние) и инструкции.

Ключевое слово может быть использован с если и пока

Ниже приведены примеры **for ... do ... end** и **while ... do/then ... end**

```
 -->n = 9;
->for k = 1:1:3 do
-->n = n - 3
-->end
 n  =
   6.
 n  =
   3.
 n  =
   0.
```

```
 -->n = 9;
-->k = 1;
-->while k <= 3
do
-->n = n -3
-->k = k + 1;
-->end

 n  =
6.
 n  =
   3.
 n  =
   0.
```

```
 -->n = 9;
-->k = 1;
->while k <= 3 then
-->n = n-3
-->k = k + 1;

-->end
 n  =

   6.
 n  =

   3.
 n  =
   0.
```

if … (elseif/else) … end

Дополнительный ElseIf и

остальное ключевые слова обеспечивающие исполнения альтернативных групп отчетности

```
если Condition_1 если
/ / Foo
ElseIf condition_2 ElseIf
/ / Foo
.....
еще
/ / Foo
конец
```

if … elseif/else … end: demo

Следующая функция вычисляет n:th член последовательности Фибоначчи, когда n дается:

```
// fibonacci.sci

// Gives the n-th term of the Fibonacci  /
// sequence 0,1,1,2,3,5,8,13,...         /

funcprot(0)            // Suppress redefenition warning
function [K] = fibonacci(n)
  if n==1              // Begin if-elseif-else-end
     K = 0;
   elseif n==2         // Condition to proceed, n > 2
     K = 1;
   elseif n>2 & int(n)==n     // Check if n is an integer >2
     K = fibonacci(n-1) + fibonacci(n-2);     // Compute
Fibonacci #
   else                // Previous conditions not met
     disp('error! -- input is not a positive integer');   // Error
message
  end                  // End of if-elseif-else-end
endfunction
```

Сохраните script, загрузите его в Scilab (на редактора), тип на консоли, имя функции с n аргумента (подсказка: не используйте большое значение!)

```
-->fibonacci(8)

ans  =

  13.
```

Проверьте, что происходит при n <1

select … case … else … end

select … case … else … end конструкция выполняет первый случай, который соответствует заявленному состоянию.

Если совпадения не будут найдены, он выполняет еще раз.

Преимущество **select … case … else … end** то, что это позволяет нам избежать многократного утверждения

выберите состояние выбора

```
случай 1
//Foo
случай 2
//Foo
.....
еще
//Foo
конец
```

*Совет: Используйте **select … case, если if …elseif … else** грозят стать слишком сложными*

*Примечание: **select … case** называется переключатель … случай в Matlab (может быть изменен в Scilab)*

select ... case ... end: demo, script

Некоторые учебники по Matlab представляет это как проблемы пьяного матроса. Он демонстрирует случайное блуждание, один фиксированный шаг за период времени

Весь процесс выполняется в одной функции (randwalk (шаги)), которая должна быть выполнена из консоли

В этом случае нет никаких проблем с наличием генератор внутри для ... ENDLOOP

В script две марки участка (O -) для каждого шага, хотя они не могут быть выделены на графике на следующей стороне

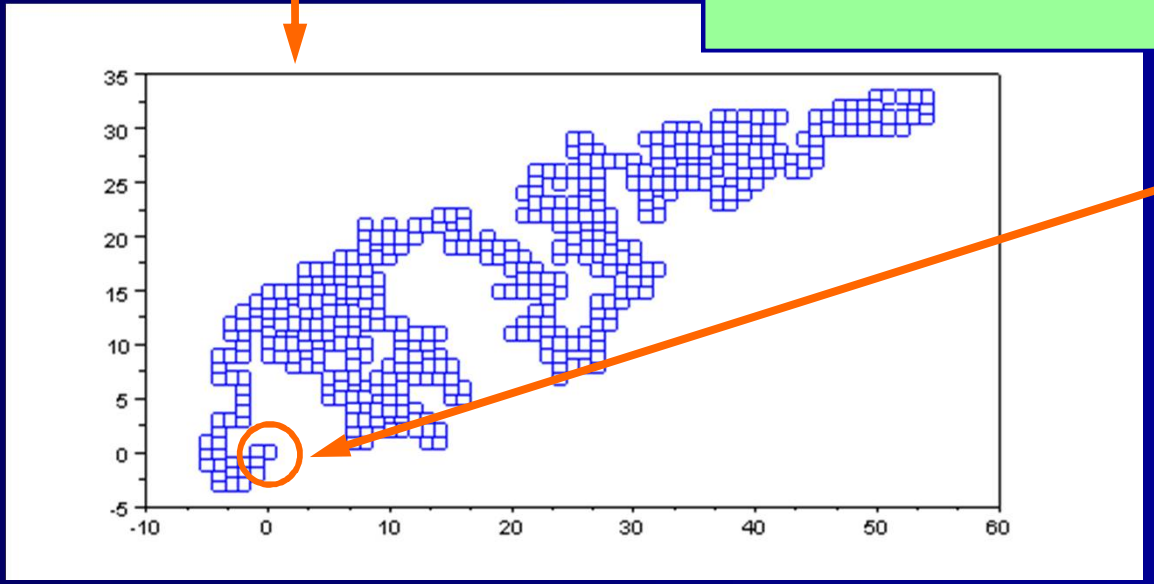
```
// randomwalk.sce
//-------------------------------------------------------
-------/
// Creates a track of marks that proceed randomly
/
// in the x,y plane. The walk starts at the origin
/
// and proceeds for a predetermined number of
steps  /
// either up, down, right, or left                    /
//-------------------------------------------------------
-------/
clear,clc,clf;
funcprot(0);
function randwalk(steps)
  x=zeros(1,steps+1);          // Counter for x track
  y=zeros(1,steps+1);          // Counter for y track
  for k=1:steps
    direction=floor(4*rand()); // Draw random
move
    select direction
    case 0 then
      x(k+1)=x(k)+1;           // Move right
      y(k+1)=y(k);
    case 1 then
      x(k+1)=x(k)-1;           // Move left
      y(k+1)=y(k);
```

select … case … end: demo, script & plot

После загрузки сценария в Scilab, функция должна быть запущена из консоли

```
-->randwalk(1000)
```

```
      case 2 then
        x(k+1)=x(k);
        y(k+1)=y(k)+1;          // Move up
      case 3 then
        x(k+1)=x(k);
        y(k+1)=y(k)-1;          // Move down
      end
    end
    clf
    plot(x,y,'o-');             // Plot marks
endfunction
```



Отправной точкой всегда является происхождение (Я делал это моделирование много раз и Scilab, кажется, предпочитает идти в северо-восточном направлении)

Перерыв: demo

```
// break.sce

// Input m positive integers that are summed  /
// up, but the program breaks if the input       /
// is not a positive integer                          /

clear,clc;

n = input('Give amount of numbers to sum_');
summa = 0;                  // Reset summa counter
for i = 1:n
   number = input('Give next number_');
   if number < 0            // Condition: number ~< 0
      disp('wrong-----negative value!');
      break;
   end
   if number ~= int(number)       // Check if
integer
      disp('wrong-----not an integer!');
      break;
   end
   summa = summa + number;    // Sum up
end
disp(['Accumulated error-free sum is:'
string(summa)]);
```

```
Give amount of numbers to sum_3

Give next number_13
Give next number_17
Give next number_7
 Accumulated error-free sum is:  37
!
```

```
Give amount of numbers to sum_3
Give next number_17
Give next number_2
Give next number_-1
  wrong-----negative value!
!Accumulated error-free sum is:  19
!
```

```
 Give amount of numbers to sum_ 4

Give next number_ 18
Give next number_ 3.3
 wrong-----not an integer!

!Accumulated error-free sum is:  18
!
```

Сломать и продолжить

Break команды:

Перерыв позволяет выйти рано из-за **... end** или в то время как ... конец цикла, или изнутри, если ... конец заявлении

Выполнение продолжается с линии, следующей за конечным заявлением

Во вложенных циклах, перерыв выходит только из самого внутреннего цикла

Команду **continue:**

продолжать -это принудительное возвращение в начале для ... конец или в то время как ... конец цикла (если не ... конец петли!)

```
-->k = 0;
-->while 1 == 1,

-->k = k + 1;
-->disp(k);

-->if k > 6 then
-->break
-->end;

-->end
   1.
   2.

   3.

   4.

   5.
   6.

   7.
```

```
 -->for j = 1:2
-->x = [];
-->for k = 1:10
-->if k>j+1 & k<=8 then

-->continue
-->end
-->x = [x,k];
-->end
-->x
-->end
 x  =
    1.    2.    9.    10.
 x  =
    1.    2.    3.    9.    10.
```

Try ... Catch ... End

При отсутствии ошибок, можно попытаться выполнить код.

Если происходит ошибка, выполнение немедленно переходит на код между catch and end:

```
try

// foo/ / Если ошибка
происходит в этой части ....

catch/ / .... Выполнение
продолжается здесь

/ / fooend
```

Обычно код между catch and end информирует об ожидаемой ошибки, например DISP ('-------- предупреждение: не может получить доступ к функции ----------')

12. Примеры ,№ 4

Первые три примера относятся к главе 10, остальные к главе 11

Пример 4-1: ступенчатая функция, блок шагом (1/2)

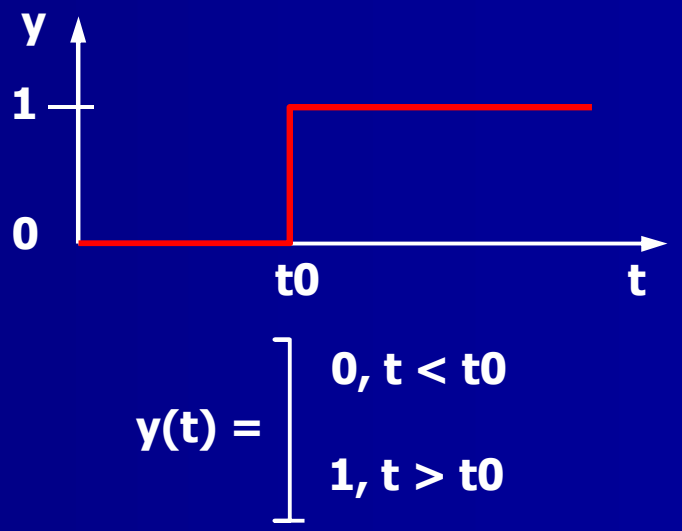
Ступенчатые функции полезны во многих практических применениях

Как уже упоминалось в примере 2-3, Scilab не хватает отдельной функции для создания (блок) шагов, но мы можем формировать их косвенно (в примере 2-3 это было сделано с единичным вектором)

Здесь мы будем смотреть на двух случаях, когда шаг необходим

В первом demo шаг будет создан пользовательской функцией, которая включает знак (функция) (помощь не поможет здесь, вы не понимаете, что он говорит о знаке ())

Unit step:

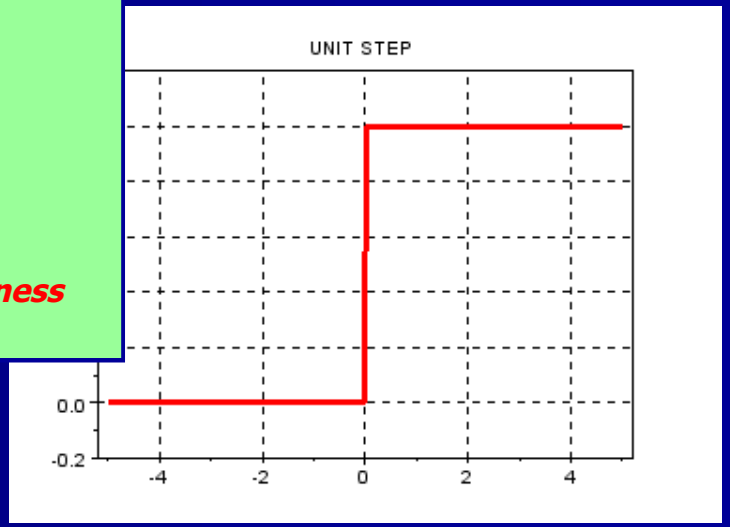


$$y(t) = \begin{cases} 0, & t < t0 \\ 1, & t > t0 \end{cases}$$

Пример 4-1: ступенчатая функция, блок шаг (2/2)

```
// step_sign.sce
// Plot a sign() fucntion that is shifted by 1 /
// & compressed by 0.5 to give a unit step /

clear,clc,clf;
funcprot(0);

x = linspace(-5,5,400);
deff('y=u(x)','y=0.5*(1+sign(x))')  // Define sign() function,
                                    // shift & compress as needed

rect = [-5.2,-0.2,5.2,1.2];         // Define plot frame
plot2d(x,u(x),5,'011',' ',rect)     // Plot inside frame
  xgrid()                           // Add grid to plot

a=gca();                            // Get axes handle
a.title.text="UNITY STEP";          // Add title
a.children.children.thickness=3;    // Increase line thickness
```

Обратите внимание, как функция **sign()** смещается дополнение к 1) и сжимается умножают на 0,5), чтобы олучить необходимый блок шаг



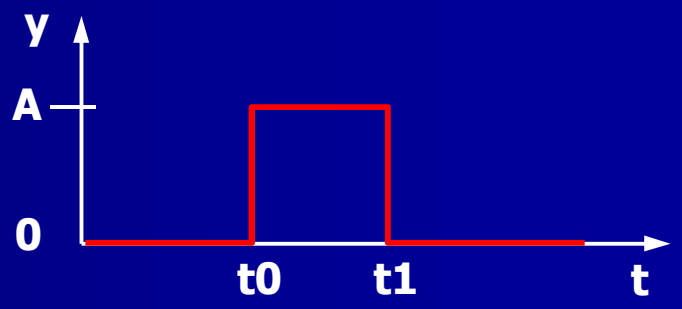
Пример 4-1: ступенчатая функция, прямоугольный импульс (1/2)

Второй случай представляет собой прямоугольный импульс с амплитудой A, как показано на рисунке

В этом случае мы делаем все без пользовательской функции, так как это приводит к более простому сценарию

Команду Plot также можно несколько упростить

Rectangular pulse:

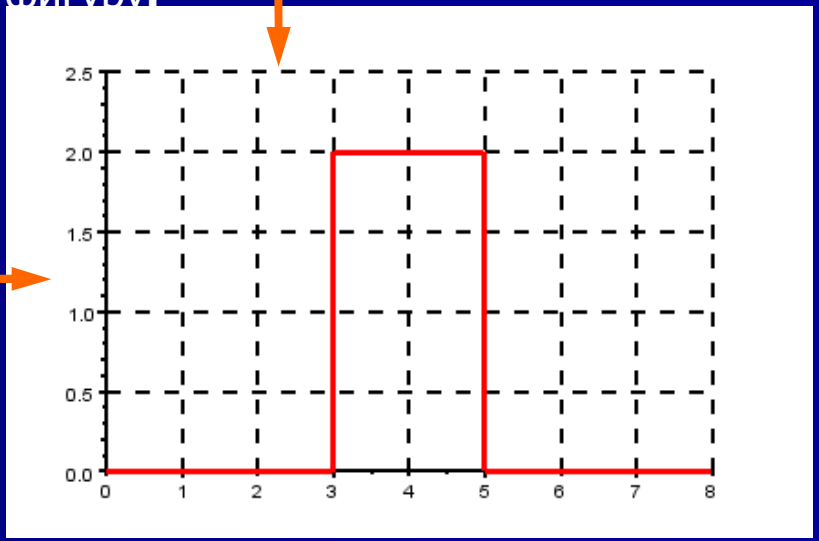


$$y(t) = \begin{cases} A, & t0 \le t < t1 \\ 0, & \text{otherwise} \end{cases}$$

Пример 4-1: ступенчатая функция, прямоугольный импульс (2/2)

```
// rect_pulse.sce

// Plot a rectangular pulse with /
// width 3 < t < 5 and amplitude 2 /

clear,clc,clf;

t = 0:0.01:10;
deff('y=u(t)','y=1*(t>=0)');     // Define u(t)

y = 2*(u(t-3) - u(t-5));         // Define pulse
plot2d(t,y,5,rect=[0,0,8,2.5])  // Plot
xgrid()                          // Add grid

f=gcf();                         // Figure handle
f.children.thickness=2;          // Figure lines
a=gca();
c=a.children;                    // Compound handle
c.children.thickness=3;          // Line thickness
```

Обратите внимание, что аргументы u () не должны быть определены отдельно
Прямоугольник можно определить, даже если стиль и STRF отсутствуют
Линии и график с повышенной толщиной линий дает жирную фигуру.



Пример 4-1: ступенчатая функция, комментарии

В первом случае (ступенчатая) команда ручка для толщины линии является

a.children.children.thickness = 3;

Во втором случае (прямоугольный импульс) Scilab не принял эту форму и это должно было быть переписано в виде

c = a.children; c.children.thickness = 3;

Я понятия не имею, почему это так, и Помощь, конечно, не поможет

В последнем случае я случайно написал script без Deff функции (), и на некоторое время все было в порядке. Но когда я добавил команды ручка, Scilab решил, что переменная y не определена. Принцип ПОЦЕЛУЙ (Keep It Simple, Stupid) не применяется в этом случае

Пример 4-2: конусы в 3D пространстве

Этот пример взят из Chancelieret соавт., Стр. 163-166

Script вполне комплексный: четыре подпрограммы и три отдельные зарисовки структуры, это делает его трудным для передачи параметров. Изменение аргумент может иметь неожиданные последствия

Некоторые аспекты сценария не были освещены и остаются без детального обсуждения здесь. (см., однако, Ex 4-3)

Объект, конус (упоминается книга и показывается ваза), строится в трех отдельных местах с помощью боковых сдвигов, поворотов и не вращательного расширения объектов.

Конусы заштрихованы с помощью ручки, которая вызывается через команду **gce()**

Функции Scilab, используемые впервые: **diag(), eval3dp(), graycolormap(), isoview(),* size()**

**) Функция IsoView () является устаревшей. Помощь Браузер рекомендует использовать frameflag = 4instead.*

Пример 4-2: script (1/4)

vertical[] рассказывает, как двигаться вдоль оси в дальнейших
расчетах. Обратите внимание на увеличение и уменьшение значения, которые
вызовут проблемы для затенения
Функция конус () генерирует конус в случае... в пример # 12 обсуждается, как это сделано

```
// cone_manipulation.sce
//
**********************************************
//
// The script generates and plots a cone with its
//
// tip at the origin. It plots two copies of the
//
// cone, one shifted and one shifted & rotated
//
//                                                    //
//
**********************************************
//
clear,clc,clf;

// Vertical reach of 3D object:
vertical=[0,1.0,1.6,2.5,2.2,2,1.6,0.9,0.5,0.3,0.3,0.4,0.6,1,1
.4,...
1.7,0,0,0.1,0.4,0.8,1.1,1.4,1.7,1.9,2.2,2.4,2.7,3,3.3,3.7,3.9
]/2;
// SUBROUTINE 1: Generation of 3D object:
//-------------------------------------------------------
function [x,y,z]=cone(reach,Z)    // Generation of a 3D
object
  x=vertical(1,Z).*cos(reach)      // Extension along x axis
  y=vertical(1,Z).*sin(reach)      // Extension along y axis
  z=vertical(1,Z).*ones(reach)     // Vertical (z) axis
endfunction
```

Пример 4-2: script (2/4)

Боковые сдвиги объектов обрабатываются в функции перевода translation()

Номера для вращения и расширение объектов является задачей homothety()

rotation() создает матрицу для поворота объектов вокруг трех осей

Таковы четыре пользовательские функции

```
// SUBROUTINE 2, Lateral shifts:
//---------------------------------------
function XYZ=translation(vect,xyz)
  XYZ=(vect(:)*ones(1,size(xyz,2)))  + xyz // Translation
vector
endfunction

// SUBROUTINE 3, Non-rotational dilation: (center =
// center of dilation, f = dilation factor)
//-------------------------------------------------------------------
----
function XYZ=homothety(center,f,xyz)
  XYZ=translation(center,diag(f)*translation(-center,xyz))
endfunction

// SUBROUTINE 4, Rotation:
//-----------------------------------
function XYZ=rotation(angle,xyz)
  angle=angle/180*%pi;       // Angle of rotation around
axes
  c=cos(angle);
  s=sin(angle);
  Rx=[1 0 0;0 c(1) s(1);0 -s(1) c(1)]      // Rotation along x
axis
  Ry=[c(2) 0 s(2);0 1 0;-s(2) 0 c(2)]      // Rotation along y
axis
  Rz=[c(3) s(3) 0;-s(3) c(3) 0;0 0 1]      // Rotation along z
axis
  XYZ=Rx*Ry*Rz*xyz
endfunction
```

Пример 4-2: script (3/4)

eval3dp () преобразует гладкую поверхность, **cone()** в композиции четырехугольных граней

Здесь мы наносим базовый конус, который имеет кончик. Внешность и интерьер конуса должна иметь разные оттенки

Объекты манипулируют векторы, созданные ранее пользовательскими функциймми.

Экс 4-2: script (2/4)

```
// ---------------------- MAIN ---------------------- //
// ---- STEP 1: CREATE & PLOT BASIC CONE ---- //

// Superimpose rectangular facets:
//-------------------------------------------
[xv,yv,zv]=eval3dp(cone,linspace(-%pi,%pi,20),1:10);
f=gcf();                    // Get Current Figure, create figure
f.color_map=graycolormap(32); // Select color

// Plot basic cone with tip at the origin:
//-------------------------------------------------
plot3d(xv,yv,zv)
e1=gce();                   // Get Current Entity handle
e1.color_mode = 24;         // Object exterior: light grey
e1.hiddencolor = 30;        // Object interior: dark grey

// ---- STEP 2: MANIPULATE & PLOT OTHER CONES ---- //

// Object manipulations parameters:
//-----------------------------------------
XYZ=[xv(:)';yv(:)';zv(:)'];          // XYZ = 3 x N matrix
XYZT=translation([1 3 -3],XYZ);      // Lateral shifts
XYZH=homothety([5 7 -3],1.5*[1 1 1],XYZT);
                                     // Non-dilational rotation
XYZR=rotation([-15 15 30],XYZT);     // Rotation
```

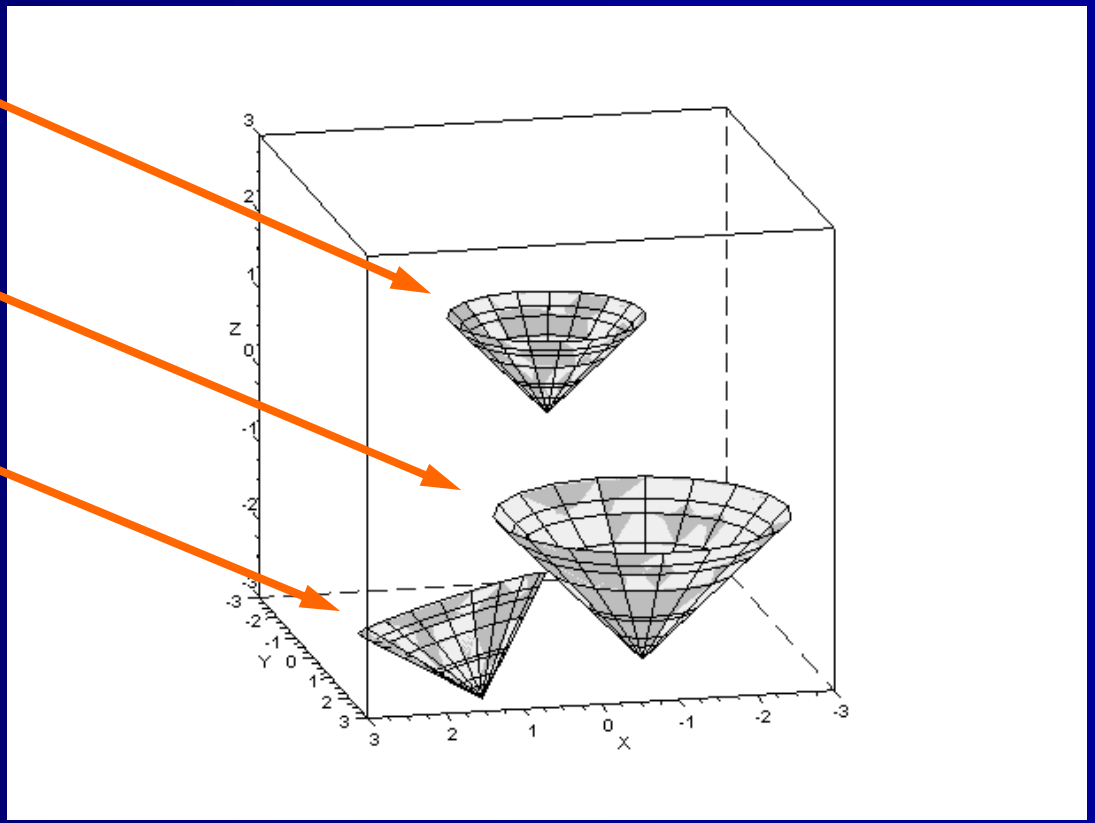
Пример 4-2: script (4/4)

Plot другого конуса, на этот раз будет увеличен и будет просвечиваться с боков. То же затенение, как и раньше
И третий plot, где конус сдвигается вбок и поворачивается. Shading как и раньше
Свойства коробки вокруг конусов корректируется. Изомет рическое масштабирование "на" (проверить в справке с объяснениями)

```
// Plot second cone (enlarged):
//-------------------------------------
plot3d(matrix(XYZH(1,:),4,-1),matrix(XYZH(2,:),
   4,-1),matrix(XYZH(3,:),4,-1))
e2=gce();                          // Get Current Entity handle
e2.color_mode = 24;                // Object exterior: light grey
e2.hiddencolor = 30;               // Object interior: dark grey

// Plot third cone (rotated):
//-----------------------------------
plot3d(matrix(XYZR(1,:),4,-1),matrix(XYZR(2,:),
   4,-1),matrix(XYZR(3,:),4,-1))
e2=gce();                          // Get Current Entity handle
e2.color_mode = 24;                // Object exterior: light grey
e2.hiddencolor = 30;               // Object interior: dark grey


// ---- STEP 3: ADJUST THE BOX ---- //

// Adjust Axes properties:
//-----------------------------
a=gca();                           // Get Current Axes
a.data_bounds=[-3 -3 -3;3 3 3];    // Box dimensions
a.rotation_angles=[75 77];         // Rotation of the box
a.isoview='on';                    // Isometric scaling

// ---- END OF MAIN ---- //
```

Пример 4-2: график

- Оригинальный конус с наконечником в начале координат
- Второй конус смещается с боков и дополняется.
- Третий конус смещается с боков и поворачивается.



И затенение все неправильное. См. Пример 4-3 для объяснения

Экс 4-2: комментарии

Chancelier др.. не задокументировали свои примеры слишком хорошо, в этом случае вместе с ошибками в их решении, вызванные основными проблемами, когда я пытался понять script. Не стоит недооценивать необходимости документирования ПРОГРАММ! Вы можете быть тем, кто страдает, когда код должен быть изменен, через ...лет после его написания

Первое требование документации-либеральное использование комментариев в коде

Среди команд некоторые, которые не были обсуждены перед: f.color_map = graycolormap, e1.color_mode, e1.hidden_color, a.rotation_angles и a.isoview = 'на' (напомним, однако команду палитры, которая была использована в примере 3-5)

Пример 4-3: как создать конус

Как генерируется конус в предыдущем примере ? Взаимодействие между матрицей вертикальной функции eval3dp [], определенной пользователем функции cone() , не слишком очевидны

Давайте упростить дело до минимума

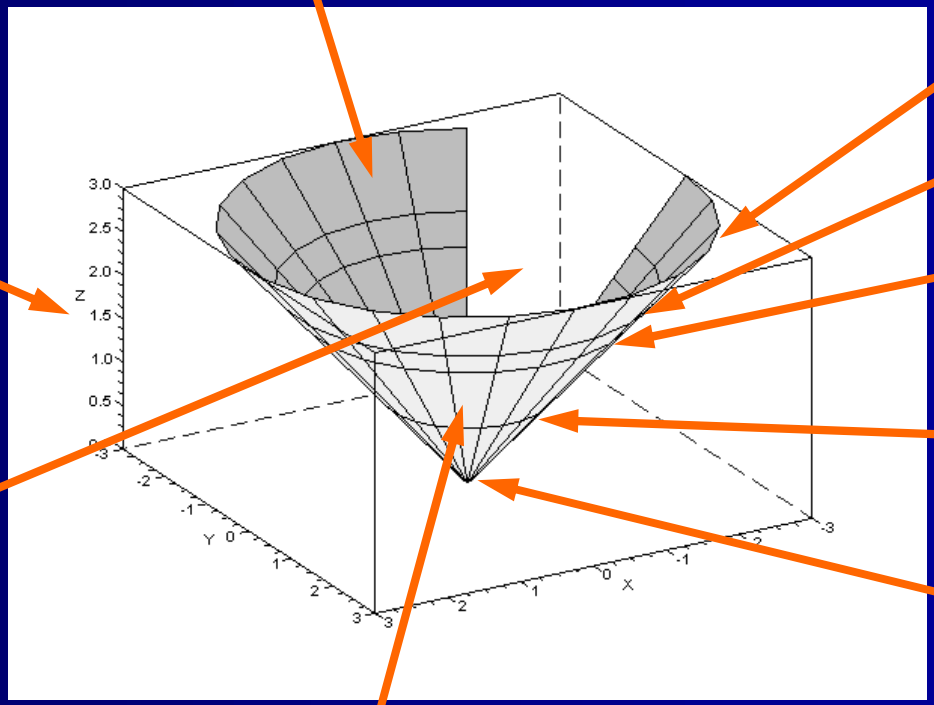
Посмотрите результат на следующем слайде

```
// cone_creation.sce

// A bare-bone eval3dp() script for plotting a 3D cone  /

clear,clc,clf;
vertical=[0,1,2,2.3,3,4];          // Vertical reach of  3D object

function [x,y,z]=cone(reach,Z)    // Generation of a 3D object
   x=vertical(1,Z).*cos(reach)      // Extension along x axis
   y=vertical(1,Z).*sin(reach)      // Extension along y axis
   z=vertical(1,Z).*ones(reach)     // Vertical (z) extension
endfunction

[xv,yv,zv]=eval3dp(cone,linspace(-%pi/1.5,%pi,20),1:5);

plot3d(xv,yv,zv,theta=60,alpha=70)   // Plot object
e1=gce();                          // Get current Entity handle
e1.color_mode = 24;                // Object exterior: light grey
e1.hiddencolor = 30;               // Object interior: dark grey
```

Пример 4-3: plot

Dark gray interior (e1.hiddencolor = 30**)**

Выравнивание Box определяется theta and alpha в plot3d ()



Z5 = 3

Z4 = 2.3

Z3 = 2

Z2 = 1

Z1 = 0

Разрыв в поверхности из-за аргумента LINSPACE (-% pi/1.5,% пи, 20)

Светло-серый внешний вид (e1.color_mode = 24)

Пример 4-3: обсуждение

Конус создается линейным увеличением радиуса RZ x и y:

$x = R_z.*\cos(Z_n)$

$y = R_z.*\sin(Z_n)$

Если вы измените первый элемент вертикали [] от 0 до 0,5, то вы увидите, что кончик конуса отрезается

Есть шесть элементов в векторе вертикали []. Последний (4) никогда не используется, так как третий аргумент в eval3dp () составляет 1:5, что означает, что только первые пять векторных элементов необходимы. Отсюда ZAXIS из участка составляет [0,3]

Я оставил пробел в периметре конуса, чтобы продемонстрировать роль второго аргумента в eval3dp ()

Этот пример имеет правильное затемнение объекта. Поверхность картины в Ex 4-2 нет художественного творчества, испортил из-за перекрытия Z_n значения

Пример 4-3: как превратить конус в вазу

Совершенно очевидно, мы должны изменить R_z в

$x=R_z.*\cos(Z_n)$

$y=R_z.*\sin(Z_n)$

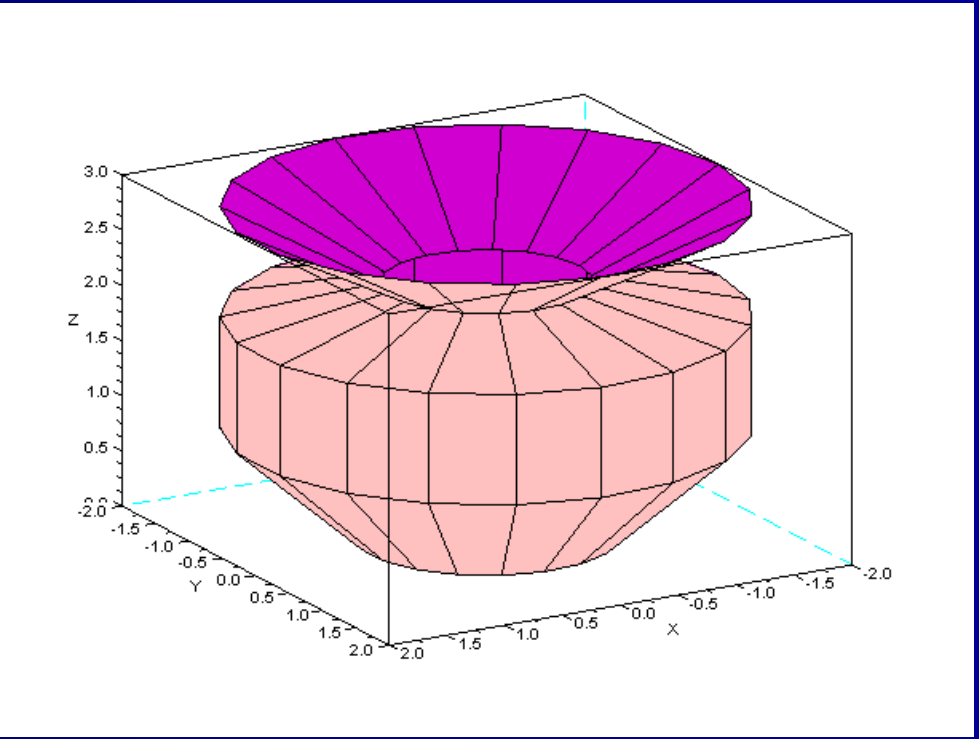
Вот один из способов это сделать: путем введения вектор R_factor, компенсирующий линейное увеличение RZ

И результат показан на следующем слайде

```
// vase_creation.sce

// A bare-bone eval3dp() script for plotting a 3D vase  /

clear,clc,clf;

vertical=[0,1,2,2.3,3,4];      // Vertical reach of 3D
object
R_factor=[1,1,0,-1.5,-1,0];      // Correction matrix

function [x,y,z]=cone(reach,Z)  // Generation of a 3D
object
  R=vertical+R_factor;           // Radius of vase, R=f(Z)
  x=R(1,Z).*cos(reach)           // Extension along x axis
  y=R(1,Z).*sin(reach)           // Extension along y axis
  z=vertical(1,Z).*ones(reach)  // Vertical (z) extension
endfunction

[xv,yv,zv]=eval3dp(cone,linspace(-%pi,%pi,20),1:5);

plot3d(xv,yv,zv,theta=60,alpha=70)   // Plot object
   e1=gce();                        // Get Current Entity handle
   e1.color_mode = 24;          // Object exterior: light
grey
   e1.hiddencolor = 30;         // Object interior: dark grey
```

Пример: ваза, график

Неплохо, а?
Но я понятия не имею, где розовые и анилиновые цвета, они ударили, когда я выполнил script, после Scilab разбился. Серая шкала вернулся после того, как я перезагрузил Scilab во второй раз



Пример 4-4: голосование двигатель для политиков

Функцией на следующих двух слайдах является машина для голосования, которая помогает политикам принять решение о том, как отдать свой голос

Ряд вопросов на голосовании вводится и код проверяет, какое число является целым положительным.

Scilab сопоставляет случайные числа и преобразует их в голоса (да / нет / воздержаться)

И наконец, голоса делят на три группы

Функция демонстрирует использование select ... case ... end

```
-->voting
Give number of issues to vote on_5
 Now this is how you should vote:
 yes
 no
 yes
```

```
-->voting
Give number of issues to vote on_-2.2
 warning-----must be > 0
```

Пример 4-4: Сценарий (1/2)

- Значительная часть из команд функций связана с проверкой достоверности данных

Первая проверка гарантирует, что число, введенное пользователем> 0

Следующая проверка, чтобы убедиться, что n является целым числом

Обратите внимание на команды прервать!

```
// voting.sci

// Ballot machine for politicians. The number   /
// of issues to be voted on is entered and      /
// Scilab tells how to vote on them. The        /
// answers are presented in groups of three     /

clear,clc;
funcprot(0)

function voting

   // Give input and check entered number:
   //-----------------------------------------------------
   n = input('Give number of issues to vote on_ ');
   if n <= 0 do                  // # of votings must be > 0
      disp('warning-----must be > 0');
      abort;
   end
   if n ~= int(n) do             // n should be an integer
      disp('warning-----not an integer!');
      abort;
  end
```

Пример 4-4: script (2/2)

Генерация случайных чисел аналогичным образом в примере 1-3

Select ... case ... end конструкция, которая преобразует случайные числа для текстовых строк Наконец, строковые выходы сгруппированы тройками. Обратите внимание, как удобна функция по **modulo()!**

```
// Create n random numbers 0,1 or 2:
//-----------------------------------------------
dt=getdate();              // Get initial seed
rand('seed',1000*dt(9)+dt(10));  // Seed random
generator
votes = floor(3*rand(n,1));  // Generate votes (0,1, or
2)

// Transform random numbers to verbal votes:
//-------------------------------------------------------
disp('Now this is how you should vote:');
for k = 1:n
  select votes(k)
    case 0 then
      disp('yes');          // 0 = yes
    case 1 then
      disp('no');           // 1 = no
    case 2 then
      disp('abstain');      // 2 = abstain
  end
  if modulo(k,3)==0     // 3 votes given?
    disp(' ')                // Leave space after 3 rows
  end
end

endfunction
```

Пример 4-4: комментарии

Scilab есть несколько команд, связанных с принудительным прекращением продолжающегося процесса: abort, break, exit, quit, return, resume. Проверьте с Помощь для деталей

В этом примере у меня возникли некоторые проблемы с прерыванием программы в правильной манере:

-Команда выхода должна завершить текущий сеанс Scilab

- Оказалось, что выход выполняется более или менее подобно команде перерыв лишь на конец нынешнего цикла

- Бросить- это грубая ошибка, которая закрывает Scilab

-Пробы и ошибки показали, что прерывание имело ожидаемый эффект прыжков в конце функции

Пример 4-5: вложенные структуры, script

Этот script содержит, **if ... elseif ... else ... end.** Конечная структуру вложенных внутри в то время **while ... end** (прочитать название для объяснения того, что делает script)

Обратите внимание, как обеспечить шкалы min() и MAX(), чтобы их пределы не превышались

while ... end

if ... end

```
// conditional.sce

// Climb up or down the scale depending on    /
// input data ("u" or "d") without exceeding   /
// the limits. The process ends when "e" is    /
// pressed                                     /

scale = [1 2 3 4 5 6 7 8 9]';      // Define scale to climb
i = 1;                             // Preset counter
strg = ' ';                        // strg = empty string
while strg ~= 'e'                  // Until the "e" key is hit
  disp(scale(i,:));                // Display location on scale
  strg = input('Exit(e), Up(u), Down(d)?','string')
  if strg == 'u' then              // If "u" is hit
     i = min(i+1, size(scale,1));  // One step up, until
highest
   elseif strg == 'd' then         // But if "d" is hit
     i = max(i-1, 1);              // One step down, until
lowest
   elseif strg == 'e' then         // If "e" is hit
     break;                        // Jump out of the loop
   else                            // Whatever else is hit
     disp('---incorrect input---')  // Present error message
  end                              // End of if statement
end                                // End of while statement
disp('you hit e=Exit')             // Exit message
```

Пример 4-5: исполнение

Я переустановил шкалу счетчика на 1 и она теперь увеличивает / уменьшает в зависимости от введенных данных.
Любой входной параметр, кроме u, d, или egive сообщает об ошибке.
Команда **break** хорошо работает в этом случае

```
1.
Exit(e), Up(u), Down(d)u
 strg  =
 u

     2.
 Exit(e), Up(u), Down(d)u
strg  =
 u

     3.
 Exit(e), Up(u), Down(d)d
strg  =
 d

     2.
 Exit(e), Up(u), Down(d)6
strg  =
 6
   ---incorrect input---

     2.
 Exit(e), Up(u), Down(d)u
strg  =
 u

     3.
 Exit(e), Up(u), Down(d)e
strg  =
 e
   you hit e=Exit
```

Домашнее задание:

Измените script с помощью select … case … else …endstructure вместо if … elseif … else … end. Какое решение проще?

13. Doing math on Scilab

- Делая математику на Scilab

Scilab содержит функции для сложной математики. Мы остановимся на более простых случаях

Математика в предыдущих главах

Глава 3: Комплексные числа, векторизованные функции, полиномы

Глава 4: тригонометрические функции, случайные функции

Глава 5: Матрицы, матричные операции, матричные операторы, символьные вычисления, генераторы случайных

Глава 6: системы линейных уравнений с действительными коэффициентами

Глава 7: 2D и 3D функции, векторные поля, гистограммы, вращение поверхности, логарифмы, полярные координаты

Глава 8: Полиномиальные выражения

Глава 9: Применение матриц и тригонометрических функций

Глава 10: Арифметика и алгебра

Глава 11: Логические выражения

Глава 12: Шаг функции, применение 3D векторных пространств

"Не беспокойтесь о ваших проблемах с математикой, уверяю вас, мои гораздо больше." Альберт Эйнштейн

Optim () и fsolve (): demo (1/4), задача

Функции Optim () и fsolv () дают нам инструменты, с помощью которых исследуем нелинейные уравнения и / или уравнения систем:

- Optim (), чтобы найти минимумы (и косвенно максимумы)
- Fsolv (), чтобы найти решения (корни) к уравнениям / системам уравнений

Optim () является довольно сложной функцией, которая появляется в запутанном описание с помощью браузера. Здесь мы будем придерживаться основного случая, применяя Optim () и fsolv () к уравнению

$$y = \sin(x) / ((x-0,1)^2 + 0,1)$$

Решается задача в два этапа:

- Во-первых, откладываем график, чтобы получить более полное представление о функции, и одновременно вычислим минимальное и максимальное значение для y с помощью Optim ()
- Тогда мы применяем fsolve () для вычисления точного местонахождения корня с помощью визуальных оценок от нанесенного графика

optim() & fsolve(): demo (2/4), script

Optim () требует Scilab подпрограмму типа [f,g,ind]=cost(x,ind).
Числовое значение grad не имеет
Построение делается с flpot2d (), которая очень похожа на plot2d ()
Я не знаю, почему там должен быть третий числовой аргумент в списке (), Scilab просто требует что-то (я попробовал и плакал...)
Второй аргумент Optim (list(),0) определяет градиент

```
// optim_list.sce

// Investigation of minima and maxima of the function  /
// sin(x)/((x-0.1)^2+0.1)                              /

clear,clc,clf;
// SUBROUTINES
//---------------------
deff('[fun1,grad,ind]=cost1(x,ind)',...          // Function
   'fun1=sin(x)/((x-0.1)^2+0.1),grad=0');
deff('[fun2,grad,ind]=cost2(x,ind)',...          // Inverse function,
   'fun2=-sin(x)/((x-0.1)^2+0.1),grad=0');  // note minus sign

// ---- MAIN ---- //
// Plot function:
//-------------------
x=-5:0.01:5;
fplot2d(x,cost1,5)                               // Plot function
xgrid

// Display min & max by calling subroutines:
//-----------------------------------------------------
disp(optim(list(NDcost,cost1,0),0))              // Display y min
disp(-optim(list(NDcost,cost2,0),0))             // Display y max
// ---- END OF MAIN ---- //
```

fsolve () и Optim (): demo (3/4)

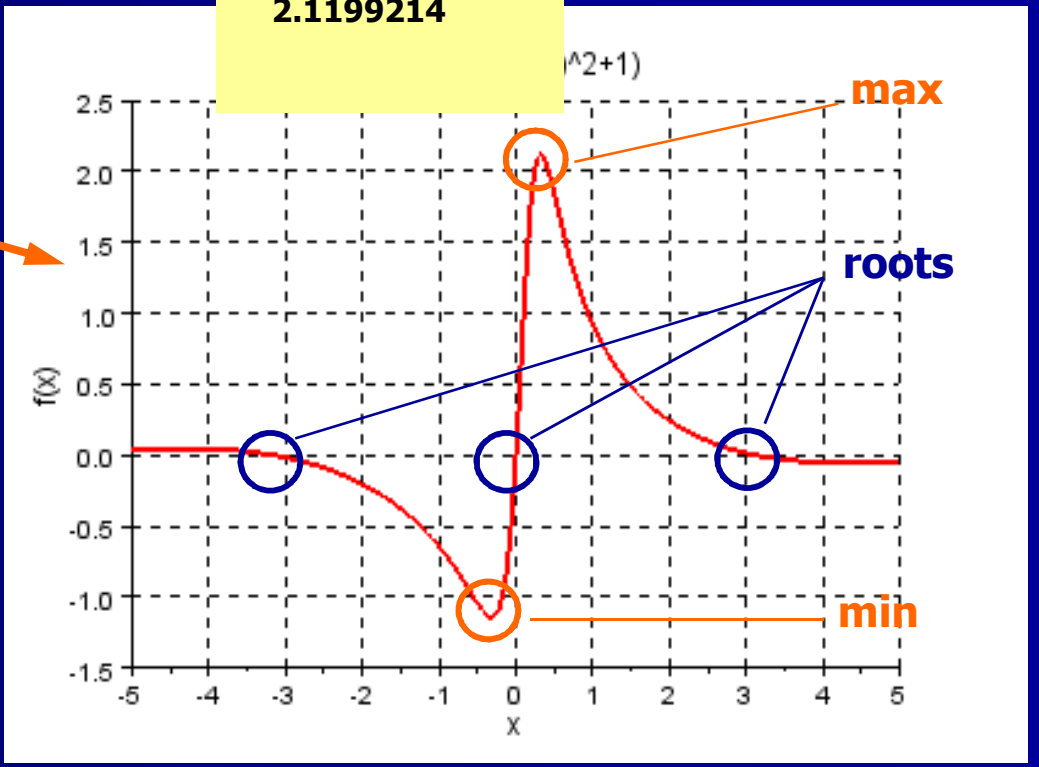
Ее минимальные и максимальные значения у произведенные Optim ()
А вот plot. Понятно, что он имеет три корня
Следующая задача -найти корни. Для этого мы должны обеспечить приближенные решения (например, -3,0,3 в данном случае), на основании которых Scilab вычисляет точное решение для данного района

- 1.1381166

2.1199214

min

max



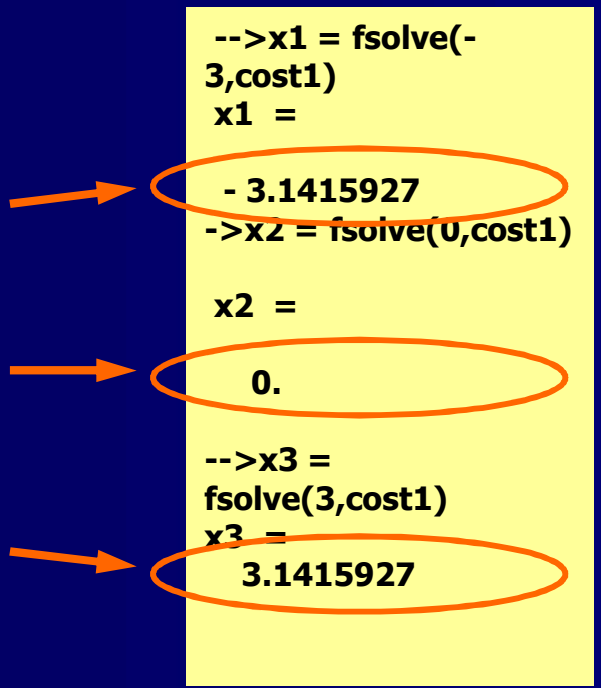
solve () и Optim (): demo (4/4), решения корни

Как сказал на предыдущем слайде, приближенные значения для корней являются:
x1 ≈ -3, x2 ≈ 0, x3 ≈ 3
Script загружается в Scilab, мы находим решения на консоли с помощью команды x = fsolve (x0, f):

x1

x2

x3



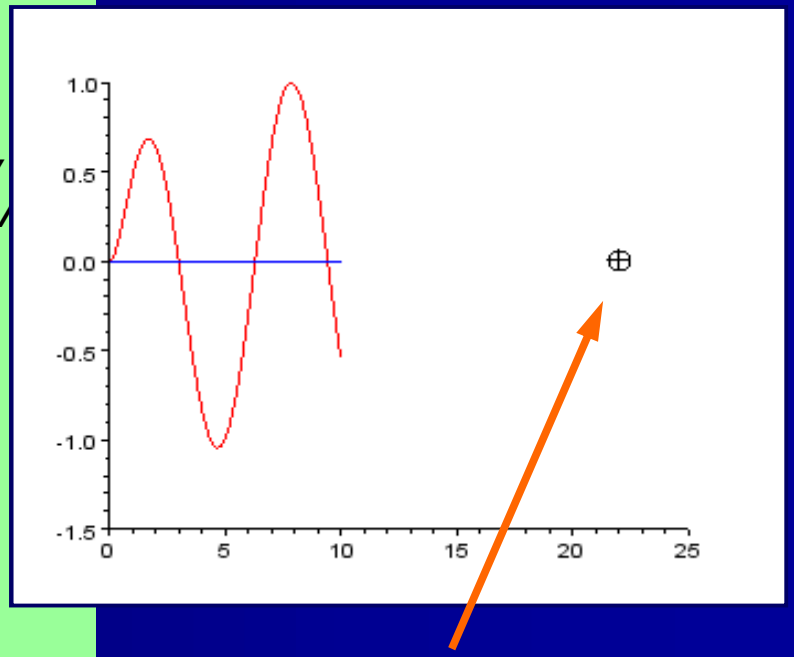
Уравнение системы требуют другого подхода. См., например, Zogg, стр. 66-69

Я уже сказал выше, что помощь Браузера смущает, когда один пытается выяснить что-то о Optim (). Лучшим источником является раздел 4.2 в Campbell u et al.

fsolve (): ограничение

Приведенный ниже script показывает, что для значений pointclose до пика недостатка кривой, например 4,6 или 8, Scilab не может решить корень правильно.

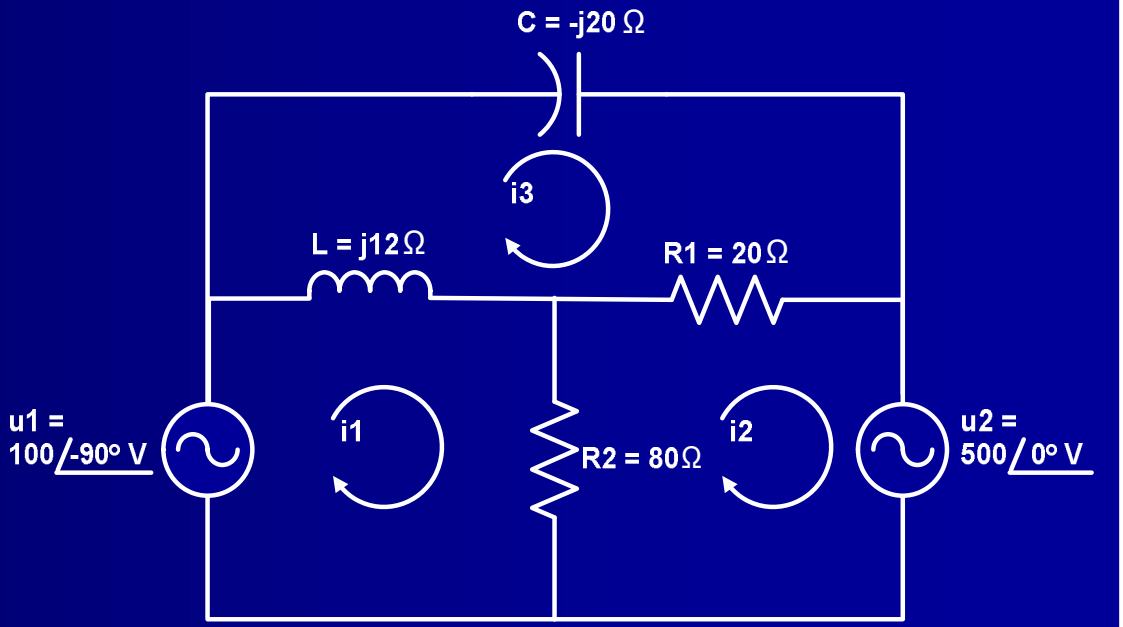
```
// fsolve.sce

// Solves, for the equation sin(a*x)-x*exp(-x),
// the root closest to a defined point.                /
//  Note: The selected point must not be too      /
// close to the midpoint between two roots          /

clear,clc,clf;
function y=myfunc(x)
  a=1;
  y=sin(a*x)-x.*exp(-x);
endfunction

x1=linspace(0,10,300);
plot2d(x1,myfunc(x1),5)     // Plot function
plot2d(x1,zeros(x1),2)      // Add y=0 graph
point = 8;                  // Point of interest
[x,y]=fsolve(point,myfunc) // Def root closest to point
plot2d(x,y,-3)              // Add mark for root location
```



Root mark in wrong place

Комплексные числа: demo, задача

- Комплексные числа не обсуждались при любой длине и раньше, так что давайте посмотрим на практическую задачу. Задача состоит в том, чтобы лишить стационарных токов I_1, I_2 и I_3 в показанной цепи.
- Напомним. Пример 2-2 и запишите полное сопротивление матриц Z осмотром



$$[Z] = \begin{bmatrix} R2+jL & -R2 & -jL \\ -R2 & R1+R2 & -R1 \\ -jL & -R1 & R1+jL-jC \end{bmatrix}$$

Комплексные числа: demo, уравнения

- При подключении, в числовых значениях получаем следующие пространства состояний. уравнение $[I]=[Z]^{-1}[u]$. Scilab не имеет функцию для перехода между полярными и прямоугольными координатами, поэтому мы пересчитываем напряжение вручную (прямоугольный к полярной, подпрограмма преобразование входит в script), оно является простым в этом случае.:

$$\begin{bmatrix} i1 \\ i2 \\ i3 \end{bmatrix} = \begin{bmatrix} 80+j12 & -80 & -j12 \\ -80 & 100 & -20 \\ -j12 & -20 & 20-j8 \end{bmatrix}^{-1} \begin{bmatrix} 0-j100 \\ -500-j0 \\ 0+j0 \end{bmatrix}$$

Обратите внимание, что u2 был выбран напротив u1, следовательно, знак минус

- Scilab. не составляет никаких проблем сделать обратные матрицы, но, как упоминалось ранее, разделение левой руки (\), как правило, дает более высокую точность.

омплексные числа: demo, script (1/3)

- Первым шагом является как Ex № 5; остаточная проверка в конце сценария
- Теперь мы должны преобразовать прямоугольные данные в полярные данные
- Для ... конец цикла проходит три раза, один раз для каждого тока (i1 ... i3)
- Расчет величины прост

```
// circuit3.sce

// Complex mesh-current solution. The complex results are
/
// converted from rectangular to polar values by computing
/
// their magnitude and phase. The clean() function is used
/
// to eliminate computing errors around zero.                    /

clear,clc;

// Compute complex currents:
//-------------------------------------
Z = [80+12*%i, -80, -12*%i;
     -80, 100, -20;
     -12*%i, -20, 20-8*%i];  // Impedance matrix
u = [-100*%i; -500; 0];      // Voltage matrix
i_n = Z\u;                   // Compute i = Z\u

// Calculate magnitude and phase:
//-----------------------------------------
magn_i = [];                 // Define empty current matrix
phase_i = [];                // Define empty phase matrix
for j = 1:1:3                // Compute for three currents
  magn_i(j) = sqrt(real(i_n(j))^2 + imag(i_n(j))^2);
                             // Computes magnitude
```

Комплексные числа: demo, script (2/3)

- Это то, где нужно быть осторожным и учитывать все варианты
- Обратите внимание, что (0) Condition Zero получает запас для вычисления ошибки через чистые функции ()
- Каждый раз, когда для … конец цикла проходит через результат матрицы (),она собирает данные.

```
// Calculate phase:
//-----------------------
if clean(real(i_n(j))) > 0 then          // In 1st or 4th quadrant
    phase_i(j) = atan(imag(i_n(j))/real(i_n(j)))*(180/%pi);
  elseif clean(real(i_n(j))) < 0         // In 2nd or 3rd quadrant
    if clean(imag(i_n(j))) > 0 then      // In 2nd quadrant
      phase_i(j) =
atan(imag(i_n(j))/real(i_n(j)))*(180/%pi) + 180;
      elseif clean(imag(i_n(j))) < 0 then     // In 3rd quadrant
      phase_i(j) =
atan(imag(i_n(j))/real(i_n(j)))*(180/%pi) - 180;
      else                                    // On negative Re-axis
      phase_i(j) = 180;
    end
  elseif clean(imag(i_n(j))) > 0       // On positive Im-axis
    phase_i(j) = 90;
  elseif clean(imag(i_n(j))) < 0       // On negative Im-axis
    phase_i(j) = -90;
  else                  // Origin: imag(i_n(j)) = real(i_n(j)) = 0
    phase_i(j) = 0;
end
result(j,:) = [i_n(j), magn_i(j), phase_i(j)];
                    // Matrix collects computed data
j = j+1;
end
```

Комплексные числа: demo, сценария (3/3) и печати

- Результат отображается с помощью команды DISP () со всем включенным в векторе аргументов
- Наконец, предварительные результаты проверяется как и раньше
- И ответ на консоли

```
// Display summary:
//-------------------------
currents = ['i1 = ', 'i2 = ', 'i3 = ']';                    // String matrix
statement = [' equals: ',' equals: ',' equals: ']';         // String matrix
disp(['CURRENTS IN COMPLEX AND POLAR FORM:'])  // Headline
disp([currents, string(result(:,1)), statement,...          // Display result
    string(result(1:3,2)), string(result(1:3,3))])
// Check residual:
//---------------------
residual = clean(u -  Z*i_n)'          // Check initial results
```

```
!
! i3 =    5-%i*25       equals:    25.495098  -7
CURRENTS IN COMPLEX AND POLAR FORM:

! i1 =   -14-%i*17      equals:    22.022716  -129.47246
!                                                      !
! i2 =   -15.2-%i*18.6  equals:    24.020824  -129.2558
!                                                8.690068    !
```

На простом английском языке:
i1 = 22.0 cos (ωt - 129,5 º)
i2 = 24.0 cos (ωt - 129,3 º)
i3 = 25,5 cos (ωt - 78,7 º)

Числовой вывод (1/3): производная ()

Производная функции f(x) определяется,

$$f'(x) = \lim_{d \to 0} \frac{f(x + d) - f(x)}{d}$$

Мы можем вычислить числовое значение f'(x) в точке x с помощью функции

производное (f(x),x,opt(d))

где opt(d) является необязательным этапом размера. Тем не менее, Scilab в помощь.

Браузер рекомендует использовать значение по умолчанию

Справа производная для theearlier исследуемой функции была рассчитана на пять различных точек.

Производное () **выводит 5x5 матрицу, в которой интерес представляют диагонали.**

```
// derivative_1.sce

// Derivative of sin(x)/((x-0.1)^2+0.1)  /
// calculated at selected points
/

clear,clc;
funcprot(0);

deff('y=f(x)','y=sin(x)./((x-0.1)^2 + 0.1)');

x = [-2 -1 0 1 2]';     // Points of interest
disp(["Point"
disp([x, diag
```

```
Point  Derivative  !


 - 2.  - 0.2800316

 - 1.  - 0.6663016

  0.    9.0909091
```

Числовой вывод (2/3): script

Этот script, где участки предыдущей функции вместе с производной

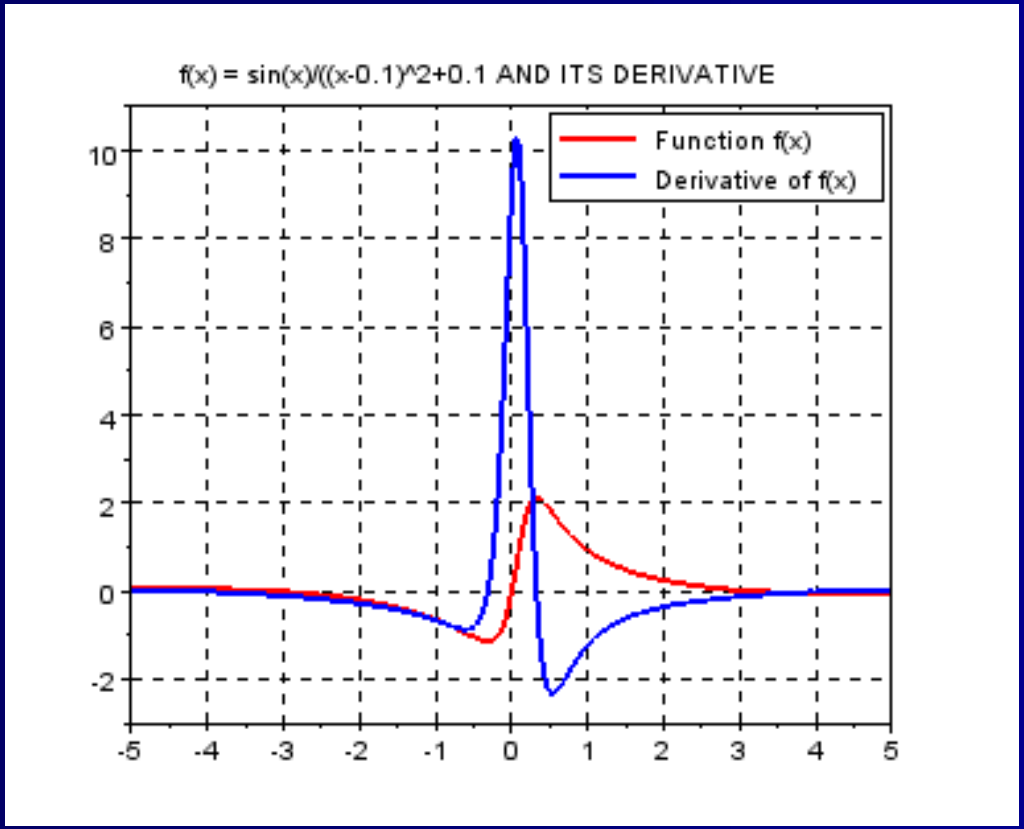
Уравнение и его производная определены отдельными Deff () функциями.

fplot2d () принимает такое же количество структур plota, который был использован ранее с plot2d ()

дети (2) и дети (3) используются, потому что дети (1) зарезервированы для легенды

```
// derivative_3.sce

// Plotting f(x) = sin(x)/((x-0.1)^2+0.1)   /
// and its derivative                       /

clear,clc,clf;
funcprot(0)

x = -5:0.01:5;                          // Area of interest
d = 0.001;                              // Step size

// Define function & derivative:
// ----------------------------
deff('y1=f(x)','y1=sin(x)./((x-0.1)^2 + 0.1)');          // f
deff('y2=g(x)','y2=((sin(x+d)./(((x+d)-0.1)^2 + 0.1))...
 -(sin(x)./((x-0.1)^2 + 0.1)))/d');                      // f'(x

// Plot function & derivative:
// --------------------------
rect = [-5,-3,5,11];
fplot2d(x,f,5,"011"," ",rect)           // Plot function
fplot2d(x,g,2,"000")                    // Plot derivative
xgrid                                   // Add grid to plot
xtitle('f(x) = sin(x)/((x-0.1)^2+0.1 AND ITS DERIVATIVE')
legend('Function f(x)','Derivative of f(x)')
a=gca();
a.children.children(2).thickness=2      // f'(x)) thickness
a.children.children(3).thickness=2      // f(x) thickness
```

Числовой вывод (3/3): plot



Урок от выполнения этого упражнения в том, что два Deff () функции в тандеме, то есть по одному на f(x), а затем по одному для f '(x), которая использует f (X), не работает. Напротив, попытка может привести к краху Scilab.

Обратите внимание ***на команду легенда в сценарии. Он поставляется до соответствующих отчетов ручки, но Scilab не жалуется.***

Численное интегрирование (1/6): определенный интеграл

Рассмотрим определенный интеграл

$$A = \int_a^b f(x)\,dx$$

Для решения интеграла, сначала определим функцию y = f (x), например, с помощью Deff функции ()

Интеграл может быть оценен с помощью INTG Scilab-функцию (), * то есть:

A= intg (a, б, e)

```
-->deff('y=f(x)',
'y=6*x^2');

-->A = intg(

 A  =

   18.
```

```
-->deff('y=f(x)',
'y=sin(x)');

-->A = intg(%pi/4,
3*%pi/4, f)
 A  =

    1.414213
```

Изменения пределов интегрирования 0 и 2 *% PI, и то, что вы получаете

```
-->A=intg(0, 2*%pi, f)

                !--error 24

Convergence problem...
```

Численное интегрирование (2/6): длина дуги

Длина дуги f(x), между точками a и б, дается определенный интеграл

$$L = \int_a^b \{1 + [f'(x)]2\}1/2 \, dx$$

Давайте вычислим длину f(x) = $x^{3/24} + 2x^{-1}$ от x = 2 x = 3
Задача требует ручного вывода.

$$f'(x) = x2/8 - 2x-2$$

```
-->deff('y=g(x)','y=sqrt(1+(x^2/8-
2*x^(-2))^2)');

-->L=intg(2,3,g)
L  =

    1.125

-->L=intg(3,4,g)
L  =

    1.7083333
```

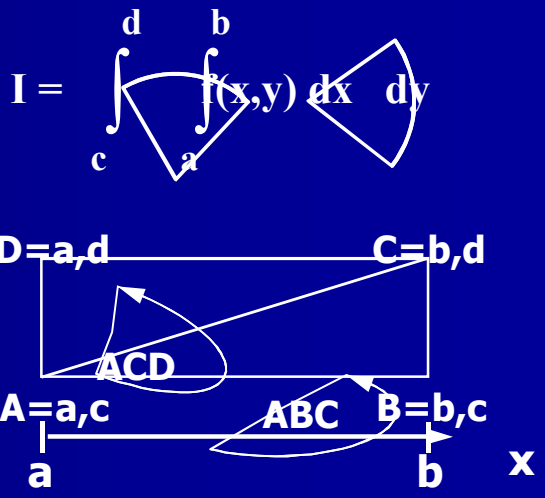
Численное интегрирование (3/6): двойной интеграл, принцип

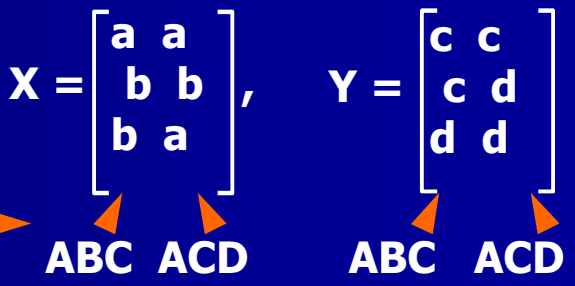
Функция int2d () вычисляет 2D область интеграла от функции f(x, y) по области, состоящей из N треугольников

x и y, следовательно, должны быть определены посредством триангуляции матриц X и Y, после чего команда [I, Err] = int2d (X, Y, f),

и Scilab возвращает переменную интегрирования I и оценку ошибки, ошибаться (не обязательно)

Треугольники ABC и ACD, как показано на рисунке. Элементы треугольника вставляются по столбцам в матрицах.





Численное интегрирование (4/6): двойной интеграл, demo

Давайте вычислим двойной интеграл

$$I = \int_{\pi/2}^{2\pi} \int_{0}^{\pi} (y\cos(x) + x\sin(y))\, dx\, dy$$

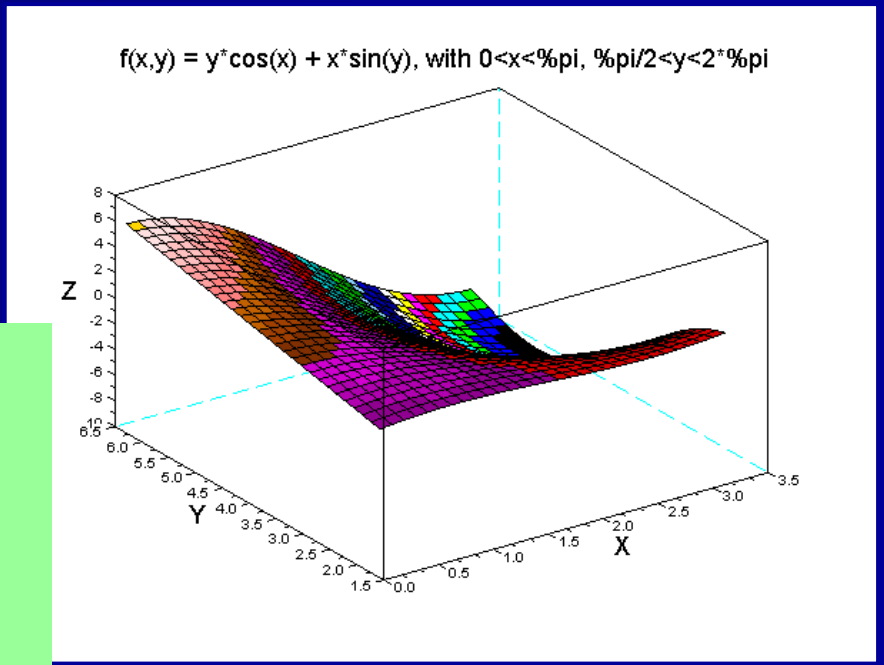
Глядя на предел s интегрирования функции мы находим триангуляции матрицы X и Y:

$$X = \begin{bmatrix} 0 & 0 \\ \pi & \pi \\ \pi & 0 \end{bmatrix}, \qquad Y = \begin{bmatrix} \pi/2 & \pi/2 \\ \pi/2 & 2\pi \\ 2\pi & 2\pi \end{bmatrix}$$

```
-->deff('z=f(x,y)', 'z=y*cos(x)+x*sin(y)');

-->X = [0  %pi  %pi;  0  %pi  0]`
 X  =

    0.          0.
    3.1415927   3.1415927
    3.1415927   0.

-->Y = [%pi/2  %pi/2  2*%pi;  %pi/2
2*%pi  2*%pi]`
 Y  =

    1.5707963   1.5707963
    1.5707963   6.2831853
    6.2831853   6.2831853

-->[I,err] = int2d(X,Y,f)
  err  =

   9.805D-11
  I  =

  - 4.9348022
```

Численное интегрирование (5/6): двойной интеграл, plot

Plot f (x, y) = y * cos (x) + x * sin(y) здесь выполняется с помощью отдельного сценария




```
// double_integral_plot.sce

// Plot the function z = y*sin(x) + x*sin(y)          /
// over the rectangle 0<x<%pi, %pi/2<y<2*%pi  /

clear,clc,clf;

x=linspace(0,%pi,30);              // Linear x axis
y=linspace(%pi/2,2*%pi,30);    // Ditto y axis
[X,Y]=meshgrid(x,y);               // Surface mesh
Z=(Y.*cos(X)+X.*sin(Y));         // 3D surface equation
surf(X,Y,Z)                              // Plot 3D surface
xtitle('f(x,y) = y*cos(x) + x*sin(y),...      // Add title
   with 0<x<%pi, %pi/2<y<2*%pi')
```

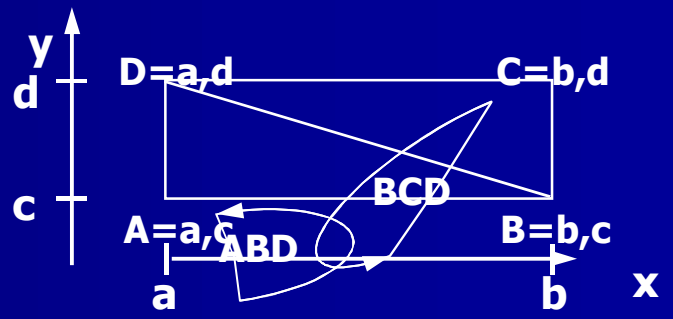
На рисунке отредактировал с рис редактора

Численное интегрирование (6/6): двойной интеграл, проверка

Мы можем проверить вычисленный результат с помощью других возможных триангуляций

Теперь мы получаем триангуляции матрицы, показанные здесь

Подключение этих матриц в к р



$$X = \begin{bmatrix} a & b \\ b & b \\ a & a \end{bmatrix}, \quad Y = \begin{bmatrix} c & c \\ c & d \\ d & d \end{bmatrix}$$

```
-->X=[0  %pi  0; %pi  %pi  0]';

-->Y=[%pi/2  %pi/2  2*%pi; %pi/2
2*%pi  2*%pi]';

-->[I,err]=int2d(X,Y,f)
err  =

    9.887D-11
I  =

  - 4.9348022
```

Тот же результат, но небольшая разница из-за расчетной ошибки

Обыкновенные дифференциальные уравнения(ОДУ): ode()*

- Простейшим требованием для решения ODE является oda (), что имеет общий вид: y = ode (y0, t0, т, e (t, y))
- где
- - y0 = начальное условие (обычно вектор-столбец)
- - t0 = начальный момент времени (обычно 0)
- - t = вектор случаях, для которых решение должно быть вычислено, например, t = [0:0.01:10
- - f (t, y) = функция, для которой решение должно быть найдено, часто указывается как [ydot] = f (t, y). Здесь t является скаляром, y вектор-столбец, и [ydot] вектор-столбец со значениями производной
- oda () также может иметь дополнительные аргументы.
- Подробности см. в справке

***) Sallet, Г.: Обыкновенные дифференциальные уравнения с Scilab, <http://www.math. унив-metz.fr / ~ Sallet / ODE_Scilab.pdf> является "старым", но хорошим текстом.**

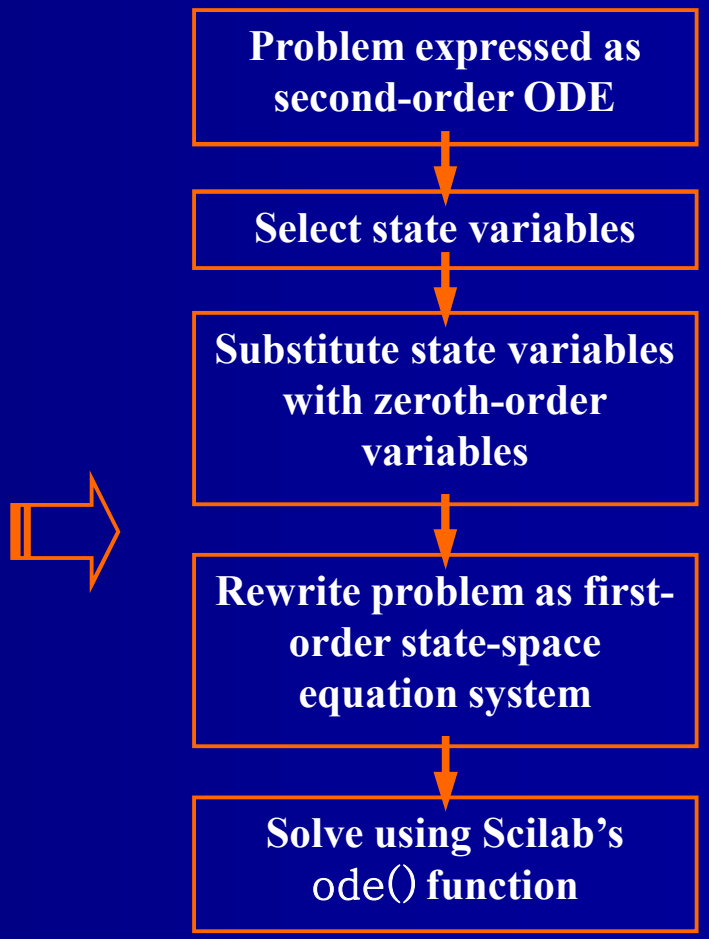
ОДУ второго порядка: введение

Scilab поддерживает только дифференциальные уравнений первого порядка-как это делают другие программы для числовых вычислений

Проблема высшего порядка должна быть уменьшена до систем первого порядка, т.е. путем перехода на пространство состояний представления

Методика работает в соответствии с алгоритмом, показанным справа

Хорошая обработка методом пространства состояний продемонстрирована, например, в главе 8 Бернс, RS: *Расширенный Control Engineering,* **Баттерворта-Heinemann, 2001**



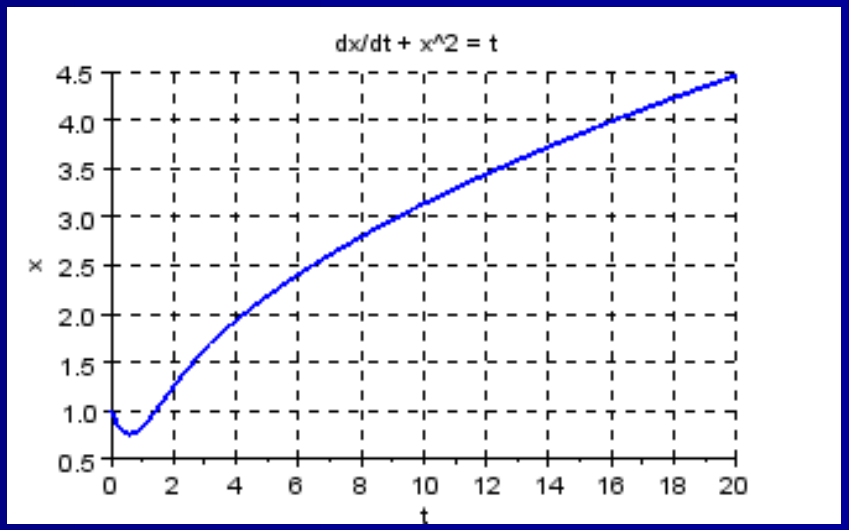
ОДУ первого порядка: demo

Давайте найдем решение для первого порядка однородной ODE $x' + x^2 = t$, с начальным условием x (0) = 1. Постройте решение для т ε [0,20]

Начните, переписав функцию x '=-x ² + t

Обратите внимание, как x 'обозначается y в *Deff* () Аргументе

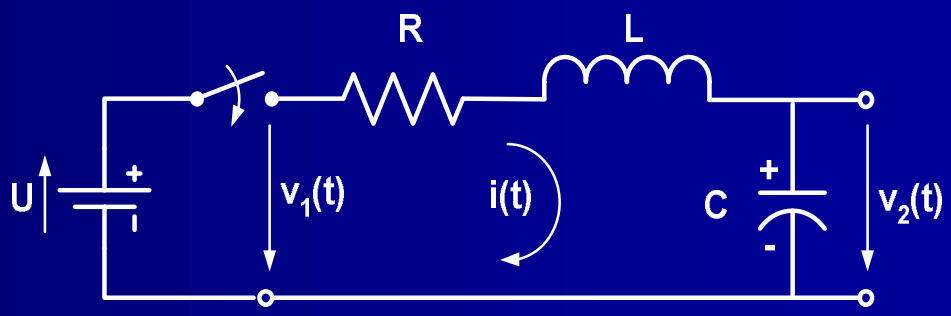
```
// first-order_ODE.sce

// Solve the equation x'+x^2 = t   /
// for x(0) = 0                    /

clear,clc,clf;
funcprot(0)

deff('y=f(t,x)','y=-x^2+t')  // Define function
t=linspace(0,20);            // Abscissa
x=ode(1,0,t,f);              // Compute equation
plot2d(t,x,style=5)          // Plot
xtitle('dx/dt + x^2 = t','t','x')
xgrid a=gca();
a.children.children.thickness=2
```



В этом случае Scilab не принимает числовые аргументы *детей*

ОДУ второго порядка: схема RLC (1/5), задача

- Задача состоит в том, чтобы построить v2 выходного напряжения для показанной цепи RLC, когда
- - U = 5 V
- - Переключатель закрывается в т = 1
- - R = 0,3 Ω
- - L = 0,5 H
- - C = 0,8 F



$$LC\frac{d2v2(t)}{dt2} + RC\frac{dv2(t)}{dt} + v2(t) = v1(t)$$

ОДУ второго порядка: схема RLC (2/5), reduce

Упрощаете уравнение для ясности:

$$LCv2'' + RCv2' + v2 = v1$$

Выбираете V$_2$ и его производную V " в качестве переменных состояния и произведите замену:

$$x1 = v2 \quad \textbf{and} \quad x2 = v2' \; (= x1')$$

V$_1$ замещена на u, первым - система ОДУ заказ будет

Это дает пространство для выражения состояний, которое мы ищем.

$$x1' = -\frac{1}{RC}x1 - \frac{R}{L}x2 + \frac{1}{LC}u$$

$$x1' = +\ 0\cdot x1 + 1\cdot x2 + 0\cdot u \qquad \textbf{(simpler: } x1' = x2\textbf{)}$$

$$\begin{bmatrix} x1' \\ x2' \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{1}{RC} & -\frac{R}{L} \end{bmatrix} \begin{bmatrix} x1 \\ x2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{LC} \end{bmatrix} u$$

ОДУ второго порядка: схема RLC (3/5), script

Напомним, обсуждение в связи с Ex 2-3: Мы работаем с матричным выражением вида

$$x' = Ax + bu$$

Все эти факторы можно видеть здесь, с x ' обозначены CC и x замещенны y.

```
// RLC_ODE.sce

// Simulation of a series RCL circuit with   /
// 5V step input voltage at t = 1s           /

clear,clc,clf;

// Define circuit components:
//-----------------------------------

R = 0.3;      // Resistance (Ohm, V/A)
L = 0.5;      // Inductance (Henry, Vs/A)
C = 0.8;      // Capacitance (Farad, As)

// Define space-state equations & input signal:
//-------------------------------------------------------
A = [0 1; -1/(L*C) -R/L];                    // System matrix
B = [0; 1/(L*C)];                            // Input matrix
deff('[ut]=u(t)','ut=2.5*(1+sign(t-1))'); // Step input signal
deff('[ss]=RLC(t,y)','ss=A*y+B*u(t)');    // Space-state expression
```

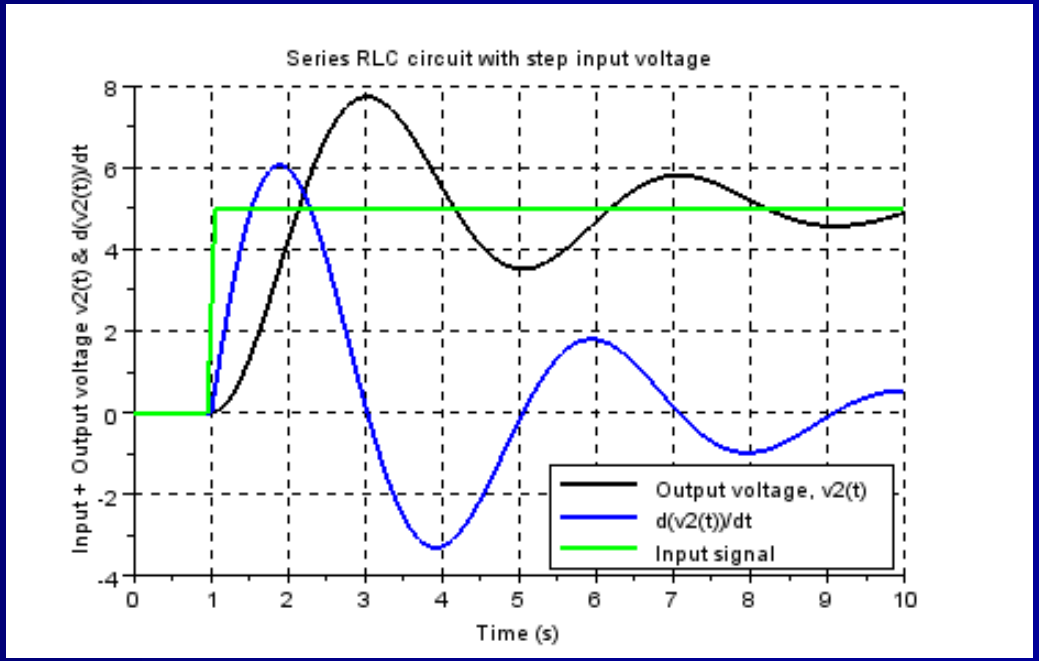
ОДУ второго порядка: схема RLC (4/5), script

Функция ода () вычисляет наше дифференциальное уравнение с помощью РЛС пространства состояний выражения второй Deff функции (). Вызов параметры 0 и t0
Обратите внимание на команду (новый способ ведения plot2d ())

```
// Compute using ode(), which calls previous deff() function:
//-------------------------------------------------------------------------
out0 = [0;0];              // Initial output voltage & d(v2)/dt = 0
t0 = 0;                    // Initial time = 0
Time = [0:0.05:10];     // Time as abscissa
State = ode(out0,t0,Time,RLC);  // State variable vector (v2',v2)

// Plot and add title & grid:
//-----------------------------------
plot2d(Time,[State',u(Time)']);    // Note transposed arguments!
   xtitle('Series RLC circuit with step input voltage',...
      'Time (s)','Input + Output voltage v2(t) & d(v2(t))/dt')
   xgrid

// Edit plot:
//--------------
a=gca();
a.children.children.thickness=2    // Make all graphs thicker

// Add legend (must come after handle commands):
//-----------------------------------------------------------------
legend('Output voltage, v2(t)','d(v2(t))/dt','Input signal',4)
```

ОДУ второго порядка: схема RLC (5/5), plot

Из графика видно, что схема *undercritically затухает.* Измените значение резистора до 1,5 Ω, и это станет *критическим.* Это *ч резмерно критично* для еще более высоких значений команд R. Ручка команды могла бы быть использована для редактирования фигуры дальше. Я не делал этого, потому что главная точка - это demонстрация решения второго порядка ОДУ.



odeoptions()

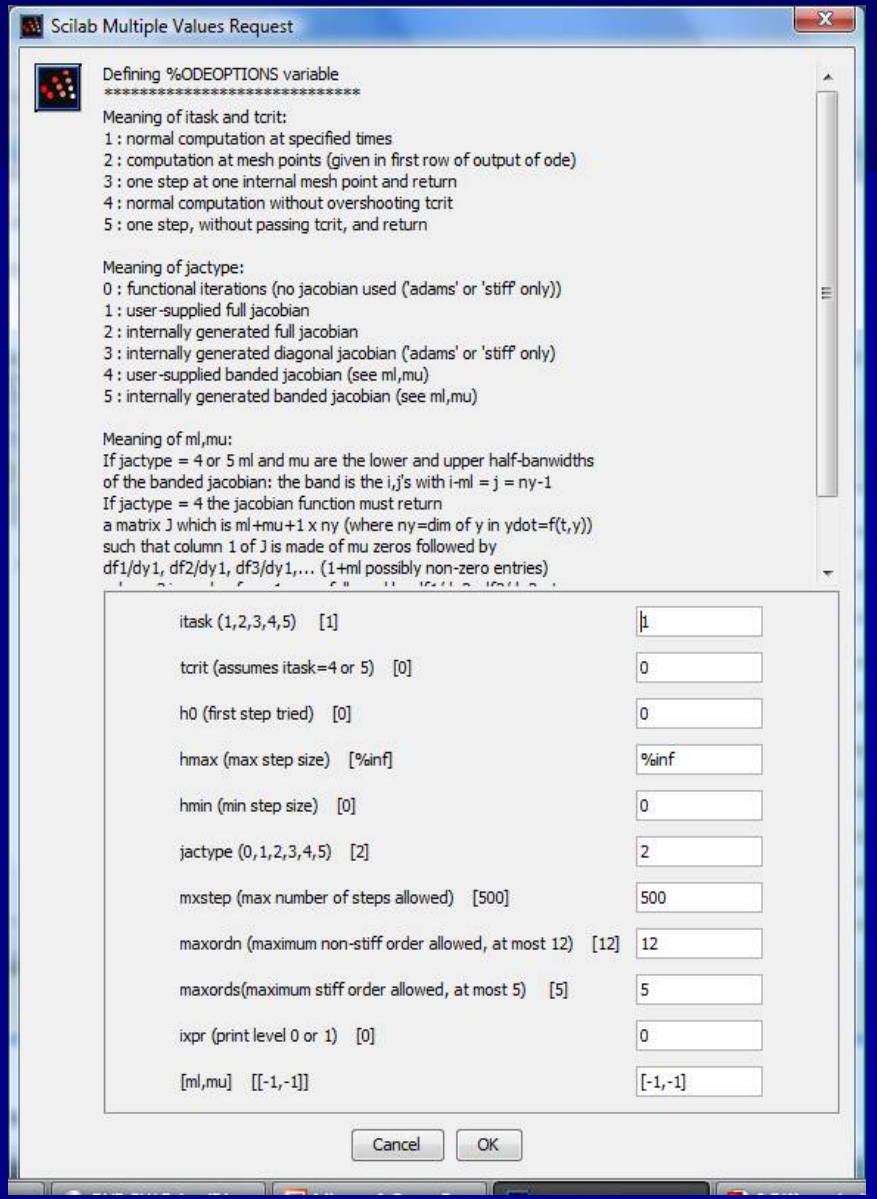
The command

%ODEOPTIONS = odeoptions()

opens the GUI shown right. With the help of it you can change parameters for solving differential equations. Examples:

- h0 **= size of first step**
- hmax **= maximum step size**
- hmin **= minimum step size**
- mxstep **= minimum # of steps**

Check with Help **for details**



Dr. EW
Johnny Heikell

14.Примеры №5

Примеры для дополнительного понимания математической работы на Scilab

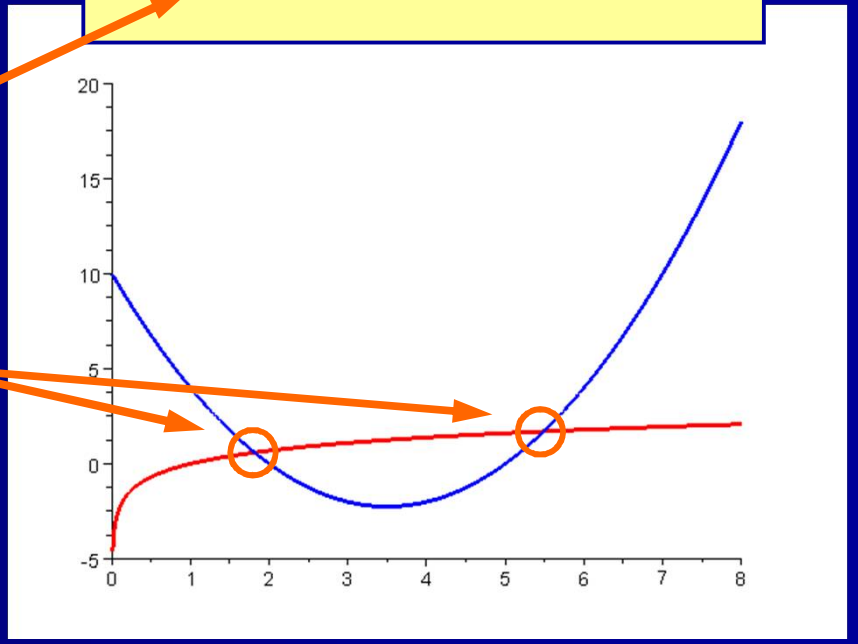
Пример 5-1: решения уравнений(1/3)

- Это демо основаное на Mäkelä
- Давайте решать уравнение:

  ln(x) = x^2-7x +10

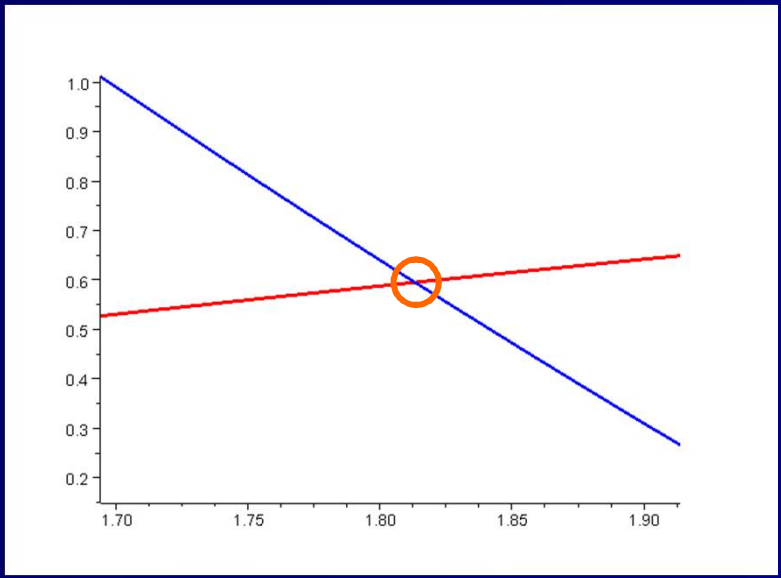
  Начнем с построения графика (обратите внимание, как построен сценарий)
  Сценарий показывает, что существуют два решения, в x1 ≈ 2 и x2 ≈ 5.5
  Вы можете точнее увидеть корни с помощью функции масштабирования графической окна (следующий слайд)

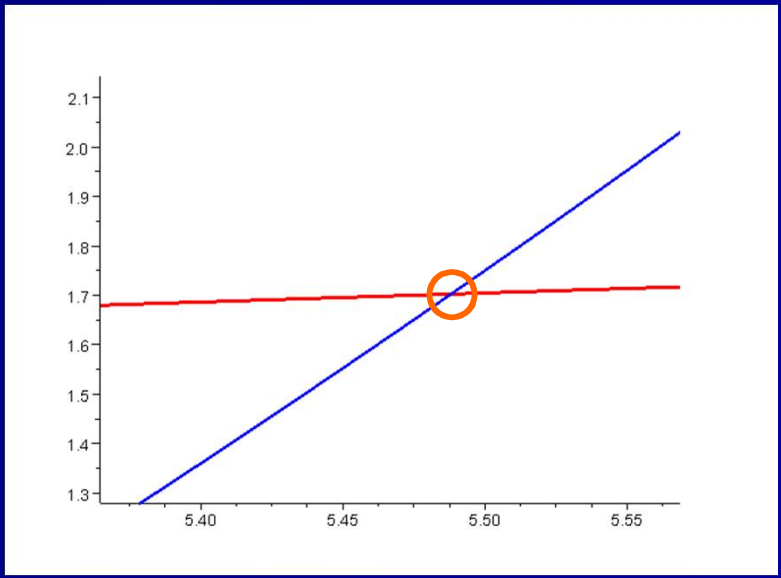
```
-->x = (0.01:0.01:8)';
-->plot2d(x,[log(x), x.^2-7*x+10])
```



Пример 5-1: решения уравнения (2/3)

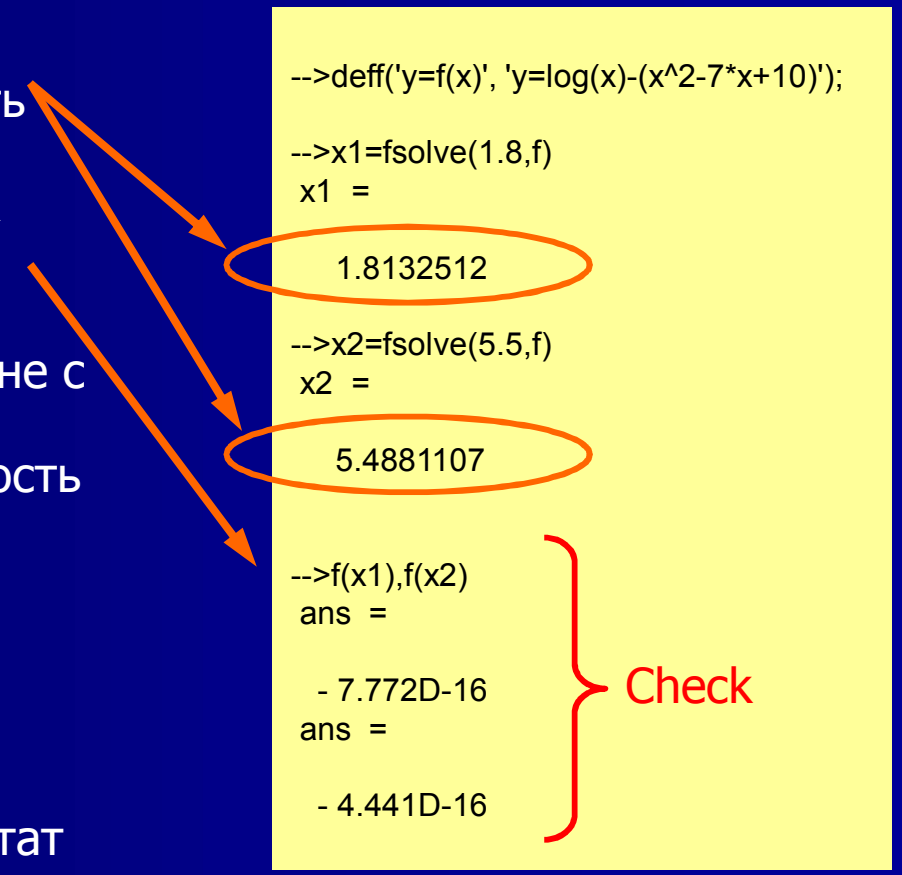
- Функция масштабирования дает более точные значения для корней: X1 = 1,81 и x2 = 5,49
- Для повышения точности еще более мы можем вычислить корни с помощью функции fsolve ()





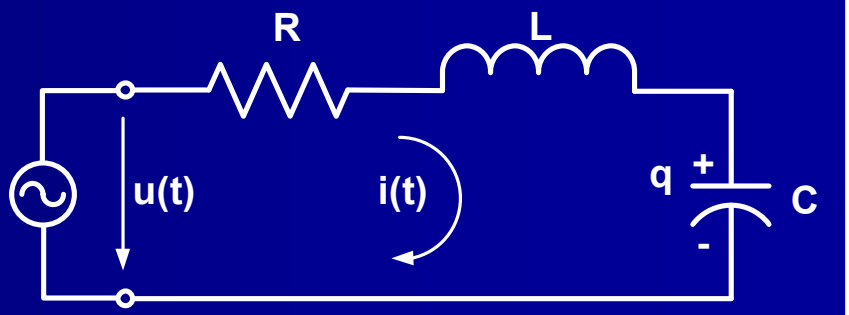
Пример 5-1: решения уравнения (3/3)

- fsolve () выводит окончательный ответ.
- Мы также можем проверить ошибку результата. Как показано, что это близко к нулю.
- Извлеченные уроки: увеличение масштаба в окне с графикой производит удовлетворительную точность для большинства практических технических целей (до десятков), учитывая, что старая инженерная пословица говорит, что факторы, которые влияют на результат менее, чем на 10% можно забыть.



Пример 5-2: ОДУ, последовательная цепь RLC (1/5)

- Этот пример является модификацией ранее цепи RLC и его второго порядка ОДУ
- Тем не менее, теперь мы хотим, чтобы определить i(t) и заряд q(t) для синусоидального входного сигнала с начальными условиями: i(0) = 0 и q(0) = 0



- Второй закон Кирхгофа дает:
- $$L\frac{di(t)}{dt} + Ri(t) + \frac{1}{C}q(t) = u(t)$$

где

$$q = \int_0^t i(t)\,dt \quad \text{или:} \quad \frac{dq}{dt} = i$$

R = 0.3 Ω
L = 0.5 H
C = 0.8 F
u(t) = sin(5t)

Пример 5-2: ОДУ, последовательная цепь RLC (2/5)

- В данном случае, никаких замен не требуется, так как q и ее производной i являются переменными состояния. Система уравнений первого порядка, следовательно:

$$\begin{cases} q' = i \\ i' = -\frac{1}{LC} q - \frac{R}{L} i + \frac{1}{L} u \end{cases}$$

Что дает следующее выражение пространствусостояний:

$$\begin{bmatrix} q' \\ i' \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{1}{LC} & -\frac{R}{L} \end{bmatrix} \begin{bmatrix} q \\ i \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{L} \end{bmatrix} u$$

запомним: $x' = Ax + Bu$

Пример 5-2: ОДУ, последовательная цепь RLC (3/5), сценарий

Здесь нет ничего нового по сравнению с предыдущим ОДУ RLC / второго порядка

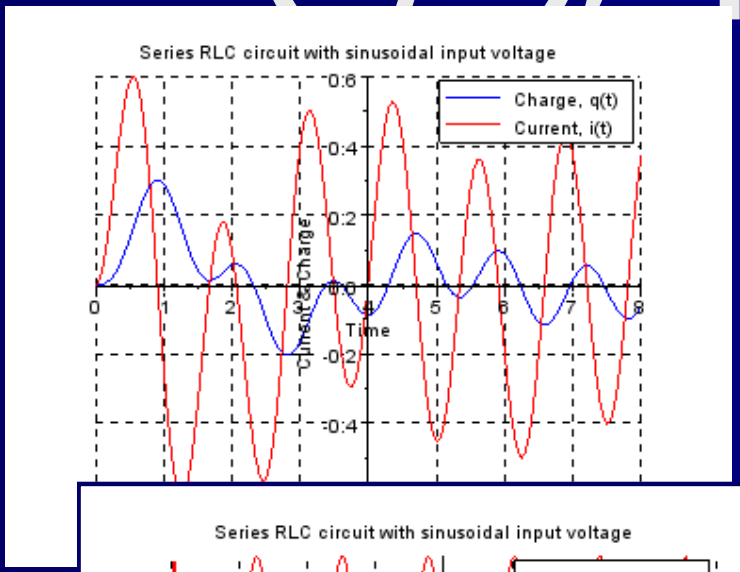
```
// series_RLC_ODE.sce

// Моделирование тока  i(t) и заряда q(t) в    /
// последовательной цепиRCL с синусоидальным входным напряжением
/
// и начальным условиеmi(0)=0, q(0)=0.                  /
// Legend: ss = state-space                              /

clear;clc,clf;

// Определение элементов схемы
//-------------------------------------
R = 0.3;                  // Сопротивление(Ом)
L = 0.5;                  // Индукция (Генри)
C = 0.8;                  // Емкость (Фарад)

// Определить пространство состояний уравнений & входной
сигнал :
//-----------------------------------------------------------
A = [0 1; -1/(L*C) -R/L];                  // SS системная матрица
B = [0; 1/L];                              // SS входная матрица
deff('[ut]=u(t)','ut=sin(5*t)');           // Синусоидальный вход
deff('[ss]=RLC(t,y)',`ss=A*y+B*u(t)');     // SS выражение
```

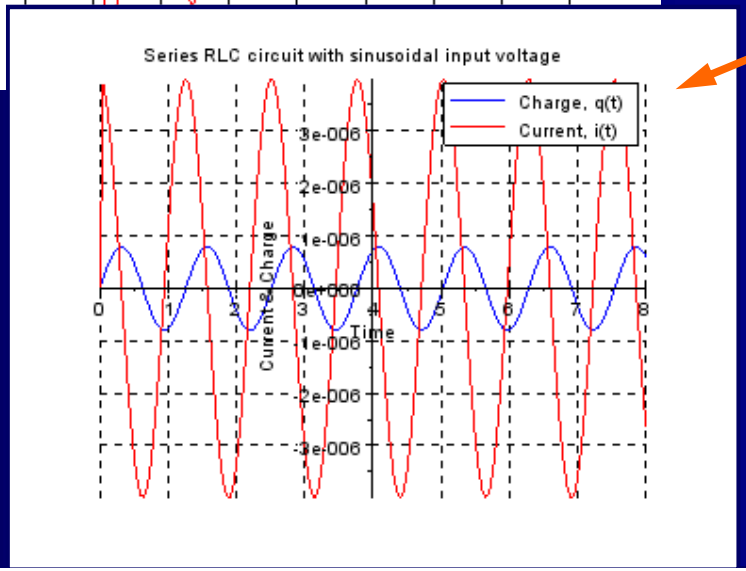
Пример 5-2: ОДУ, последовательная цепь RLC (4/5), сценарий

ode() такая же, как и в предыдущем случае RLC
Проверьте участок аргумент '024 'и его влияние на участке (следующий слайд)

```
// Вычислить с помощью ode(), которая призывает предыдущую функцию deff() :
//-------------------------------------------------------------------------------------
y0 = [0;0];                    // Начальный ток и заряд = 0
t0 = 0;                        // Начальное время = 0
Time = [0:0.05:8];             // Время как абсцисса
Y = ode(y0,t0,Time,RLC);       // Y =  вектор состояния(i,q)

// График тока и заряда:
//-----------------------------
plot2d(Time,Y',[2 5],'024');    // график основного вектора, примечание транспонирование  Y
   xtitle('Series RLC circuit with sinusoidal input voltage',...
      'Time','Current & Charge')
   xgrid
   legend('Charge, q(t)','Current, i(t)')
```

Пример 5-2: ОДУ, последовательная цепь RLC (5/5), график





- Это участок показывает значения компонентов. Есть начальная функция, до стабилизирования.
- Это участок более реалистичных значений компонент R = 3 кОм, L = 0,5 мкГн, C = 0,8 мкФ
- Могут быть проблемы в последнем случае.

Пример 5-3: Система первого порядка ОДУ

- Этот пример редактировался повы (стр. 66-67, повы также имеет анимационную версию на стр. 67-68, но это вызывает Scilab к краху). Пример заканчивается и команда интересная plot2d ()

Задача состоит в том, чтобы построить по склону (вектор) поля для следующей системы первого порядка

- :

$$\begin{cases} x' = y \\ y' = -x - y \end{cases} \Rightarrow \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

вместе с первоначальной траектории $x(0) = 1$ и $y(0) = 1$

Сценарий может использовать либо ODE систему (как повы сделал) или представление в пространстве состояний. Мы выберем последний, в соответствии с более ранних примерах

Пример 5-3 сценарий

В пространстве состояний функция названа firstorder ()

Векторное поле обращается с fchamp ()

ODE= () имеет только один аргумент (и принимает только одно имя) при начальном условии · x и y переименован x (1) и x (2) соответственно, как показано в аргументах для plot2d ()

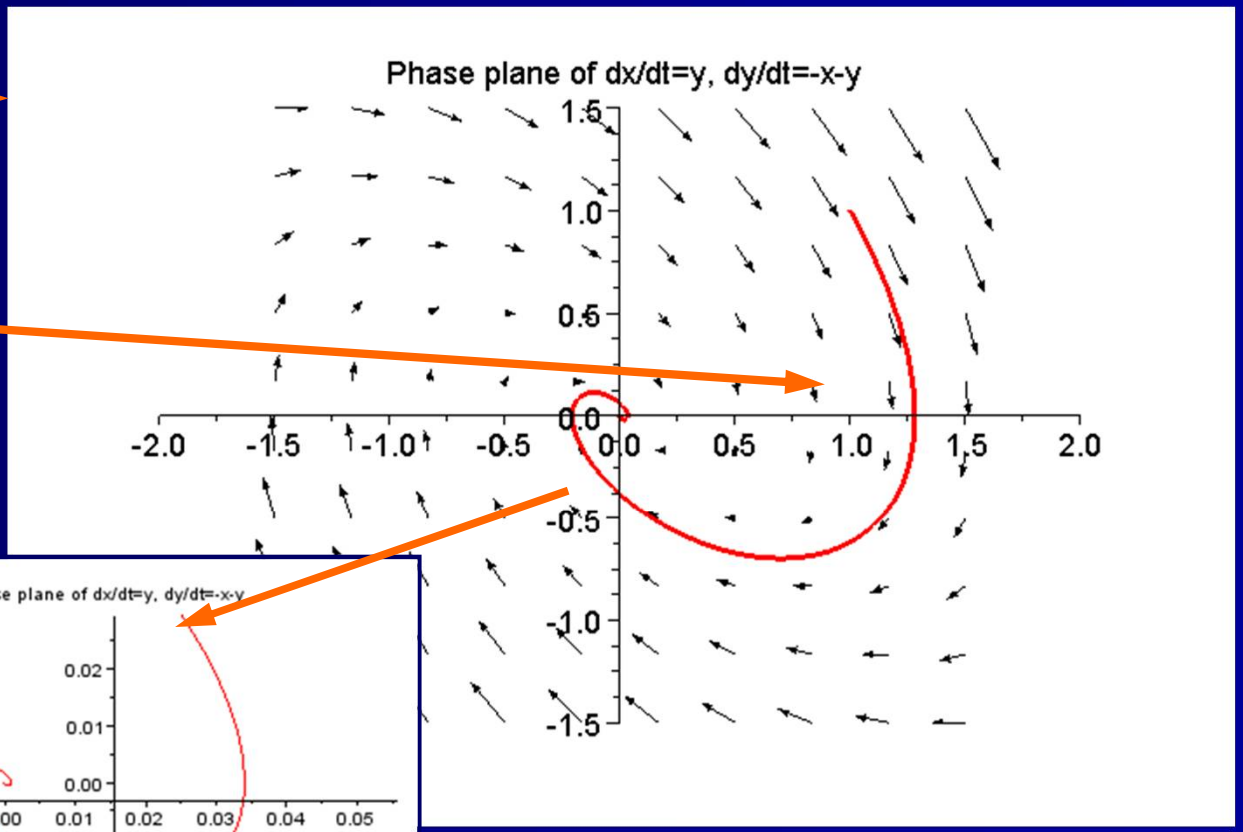
```
// ode_phase_plane_m.sce

// The scripts plot the phase plane of the          /
// equation system x'=y, y'=-x-y together with     /
// a single phase portrait that satisfies the       /
// initial condition x(0)=1, y(0)=1                  /

clear,clc,clf;
funcprot(0);

// First order transformation:
//-----------------------------------
A = [0 1;-1 -1];                  // State vector
deff('[ss]=firstorder(t,x)',`ss=A*x');

// Create & draw slope (vector) field:
//-------------------------------------------------
z = linspace(-1.5,1.5,10);
fchamp(firstorder,0,z,z)        // Draw vector field

// Create phase portrait:
//------------------------------
x0 = [1;1];                       // Initial condition
t = linspace(0,30,300);
[x] = ode(x0,0,t,firstorder);  // [x]=state variable vector
                                  // with x=x(1), y=x(2)

// Plot phase portrait on slope field:
//-------------------------------------------
plot2d(x(1,:),x(2,:),5,'004')
   xtitle('Phase plane of dx/dt=y, dy/dt=-x-y')
```

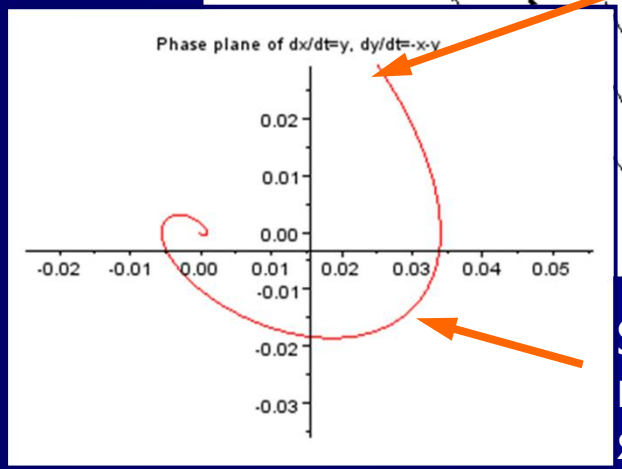
Пример 5-3: график

Полный сюжет

Фазовый портрет с начальным условием [1,1]

Увеличенный центр площадью





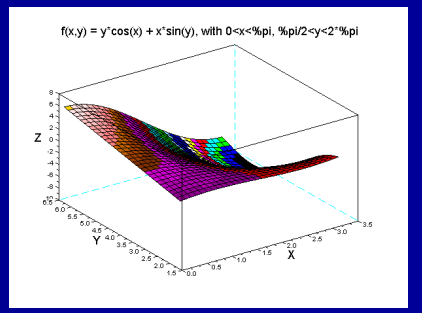
Scilab не поставил "haircross" в начало координат, которая является успехом

Пример 5-4: Simpson's задача

Этот пример демонстрирует интеграцию двойных интегралов по правилу Симпсона для вычисления двойных интегралов

- Давайте сначала зададим подпрограмму для правила Симпсона. А затем добавте код для функции, интеграла площадей, из которых должна быть рассчитана
- В этом случае мы будем повторять функцию.

$$I = \int_{\pi/2}^{2\pi} \int_{0}^{\pi} (y \cos(x) + x \sin(y))\, dx\, dy \ ,$$



но сценарий легко может быть изменен для других алгебраических выражений

- Есть множество вариантов правила Симпсона для двойных интегралов (для точного алгоритма, см. Faires, бремя: Численные методы, 3-е изд, Брукс Коул 2002 года.). Вариант, используемый на следующем слайде основан на Urroz и известен как 1/9 правила Симпсона.

Пример 5-4: правило Симпсона, алгоритм

$$I = \frac{\Delta x\, \Delta y}{9} \sum_{\substack{i=2 \\ i=i+2}}^{n} \sum_{\substack{j=2 \\ j=j+2}}^{m} S_{i\,j}$$

где мы рассчитываем нашу функцию F (x, y) в прямоугольной области R = {a<x<b, c<y<d}

Здесь x делится на n и y на т даже части, так что :

$$\Delta x = \frac{b - a}{n}, \qquad \Delta y = \frac{d - c}{m}$$

Кроме того:

$$S_{i\,j} = f_{i-1\,j-1} + f_{i-1\,j+1} + f_{i+1\,j-1} + f_{i+1\,j+1} + 4(f_{i-1\,j} + f_{i\,j-1} + f_{i+1\,j} + f_{i\,j+1}) + 16\, f_{i\,j}$$

Пример 5-4: Симпсона, скрипт

Скрипт построен в четыре этапа:

1.Общий заголовок комментарии для программы

2. Декларация UDF с последующим разъяснением комментариями

3. Свод UDF (следующий слайд)

4. Код для F (x, y), что вызывает UDF (два слайда вниз)

```
// double_integration_simpson.sce

//-----------------------------------------------------------------/
// The program calculates the double integral of the     /
// function f(x,y) = y*cos(x)+x*sin(y);  by calling the   /
// subroutine simpson_double(x0,xn,n,y0,ym,m,f)        /
//-----------------------------------------------------------------/

clear,clc;

function [integral] = simpson_double(x0,xn,n,y0,ym,m,f)

  // The function calculates the double integral of        /
  // the function f(x,y) in the region x0<x<xn,            /
  // y0<y<ym using Simpson's 1/9 rule. The x- and      /
  // y- ranges are divided into n and m subintervals,    /
  // respectively, where both m and n must be even.   /
  // The function modifies m and n if they are odd       /
```

Пример 5-4: Симпсона, скрипт

Это тело пользовательской функции
Оно начинается с проверки и (при необходимости) коррекции входных параметров n и m

Здесь мы снова встречаем функцию feval (). Она возвращает матрицу z(i,j) = f(x(i),y(j))

Сердце UDF: двойная сумма, которая производит Sij до формирования окончательного ответа (выходной аргумент)

```
// Check that n and m are even, correct as needed:
//-----------------------------------------------------------------
if modulo(n,2) <> 0 then     // Check that n is even;
  n = n + 1                  // if not, add one
end
if modulo(m,2) <> 0 then     // Check that m is even;
  m = m + 1                  // if not, add one
end

// Define x and y increments and region:
//-----------------------------------------------------
Dx = (xn-x0)/n               // Define delta x
Dy = (ym-y0)/m               // Define delta y
x=[x0:Dx:xn]                 // Region and increments of x
y=[y0:Dy:ym]                 // Region and increments of y

// Calculate double integral:
//-----------------------------------
z=feval(x,y,f)               // Matrix z(i,j)=f(x(i),y(j))
Sij = 0                      // Initiate Sij
for i = 2:2:n                // Sum Sij along x-axis
  for j = 2:2:m              // Sum Sij along y-axis
    Sij = Sij + z(i-1,j-1)+z(i-1,j+1)+z(i+1,j-1)+z(i+1,j+1)...
    +4*(z(i-1,j)+z(i,j-1)+z(i,j+1)+z(i+1,j))+16*z(i,j)
  end
end
integral = (Dx*Dy/9)* Sij    // Evaluate integral

endfunction
```

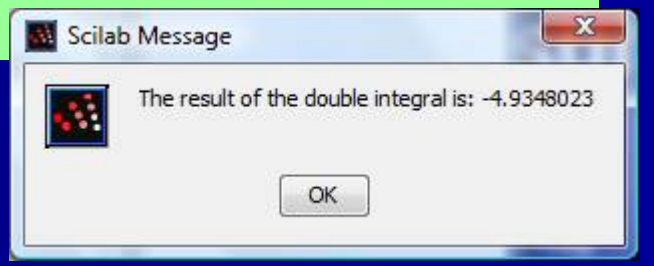
Пример 5-4: Правило Симпсона, скрипт и результат

Сейчас идет функция f (x, y), что мы хотим интегрировать. Мы начнем с определения пределов интегрирования и шагов

Интересная проблема: Как следует определить призвание аргумент F (x, y) Если вводится как F (x, y) = y * cos (x) + x * sin (y), Scilab будет жаловаться, что x и y не определены. Решение deff()

И, наконец: ответ, как показано в окне сообщения (последняя цифра раннего демо была более точным)

```
// Define integration parameters:
//-----------------------------------------
x0 = 0;                  // Lower bound for x
xn = %pi;                // Upper bound for x
n = 100;                 // # of subintervals of x
y0 = %pi/2;              // Lower bound for y
ym = 2*%pi;              // Upper bound for y
m = 100;                 // # of subintervals of y

// Define function & calculate integral:
//-----------------------------------------------
deff('[z]=f(x,y)','z = y*cos(x)+x*sin(y)');
I = simpson_double(x0,xn,n,y0,ym,m,f)
messagebox('The result of the double integral is:...
    '+string(I))
```



Точный ответ является $-\pi^2/2 = -4.934802199...$

Пример 5-4: Правило Симпсона, обсуждение

- У меня были большие проблемы с этим. Scilab неоднократно настаивал выдавал неправильный ответ. Я пытался найти ошибку в нескольких направлениях:
- Проверено вручную, какой ответ был правильным (- 4,9348022, или- п^2 / 2)
- Изменил тригонометрические функции в показательных эквивалентах, но не дало никаких результатов
- Проверил алгоритм путем сравнения решаемых примеров из области математики и Matlab в книгах

Наконец, когда я включил в уравнение в настоящее время несколько раз менял скрипт, результат вышел прямо. Скорее всего я написал sin(x) вместо cos(x).....

- Накопленный опыт: Трудно видеть ошибки в собственной программе
- Другое дело: Скрипт использует вложенные циклы (for i = ...; for j = ...; ... end; end;). Этого следует избегать в Scilab, насколько это возможно, потому что производительность в таких случаях оставляет желать лучшего

Dr. EW
Johnny Heikell

15. Работа с графическими интерфейсами

Термин GUI относится к встроенным окнам Scilab и для определенния пользователем интерактивных окон

Введение

- Графический интерфейс Scilab был обновлен с версии 5. Старые учебники (например, Кэмпбелл и др..), ограничивает использование информации
- Краткие обсуждения графического интерфейса можно найти в Kubitzki и в Антонелли и Chiaverini (вы можете прочитать сценарии Scilab на немецком и итальянском, даже если вы не говорите на языке)
- Ранее мы уже видели случаи с диалоговом окном (x_dialog () в примерах. 1-3) и MessageBox (MessageBox () в примерах. 5-4)
- Первое обсуждение ниже о том, как адаптировать окна Scilab
- После этого мы будем смотреть на некоторые диалоговые окна, определяемых пользователем. "Реальный" GUI представленный в примере 6-1

Прошив окна (1/2)

Существуют четыре основных функции для прошива либо консоли или графического окна:

Addmenu (<gwin>, кнопка, <optional arguments>)	добавляет новые кнопки или меню в основной и / или Графиком окное и командной панели
delmenu()	удаляет кнопки или меню от addmenu ()
setmenu()	активирует кнопки или меню от addmenu ()
unsetmenu()	деактивирует кнопки или меню от addmenu ()

- Если присутствует числовой аргумент gwin, то он , говорит, на каком Графиком Окне кнопка должна быть установлена
- Аргумент button строчной символов, который определяет ярлык в строке меню

Прошив окна (2/3)

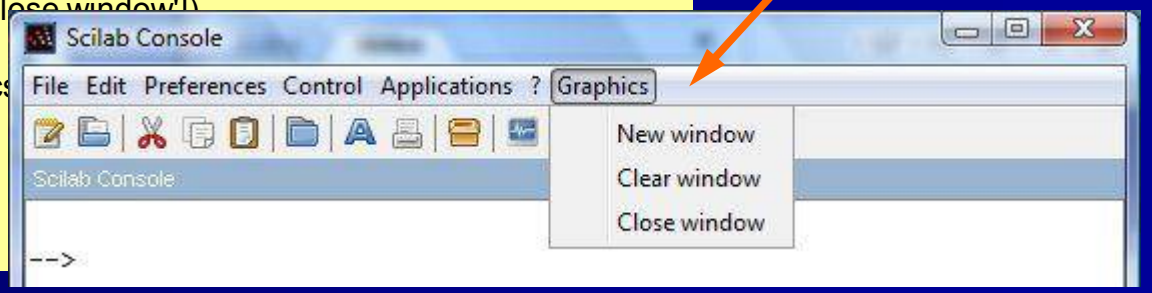
Дополнительные аргументы:

– подменю символьная строка с именами подпунктов

- Действие списка определяется типом Action = списке (флаг, проц. название)
- Это не вся правда. Книга Дас, которая представляет собой набор из Помощи функциональных текстов Scilab, в содержит несколько подсказок

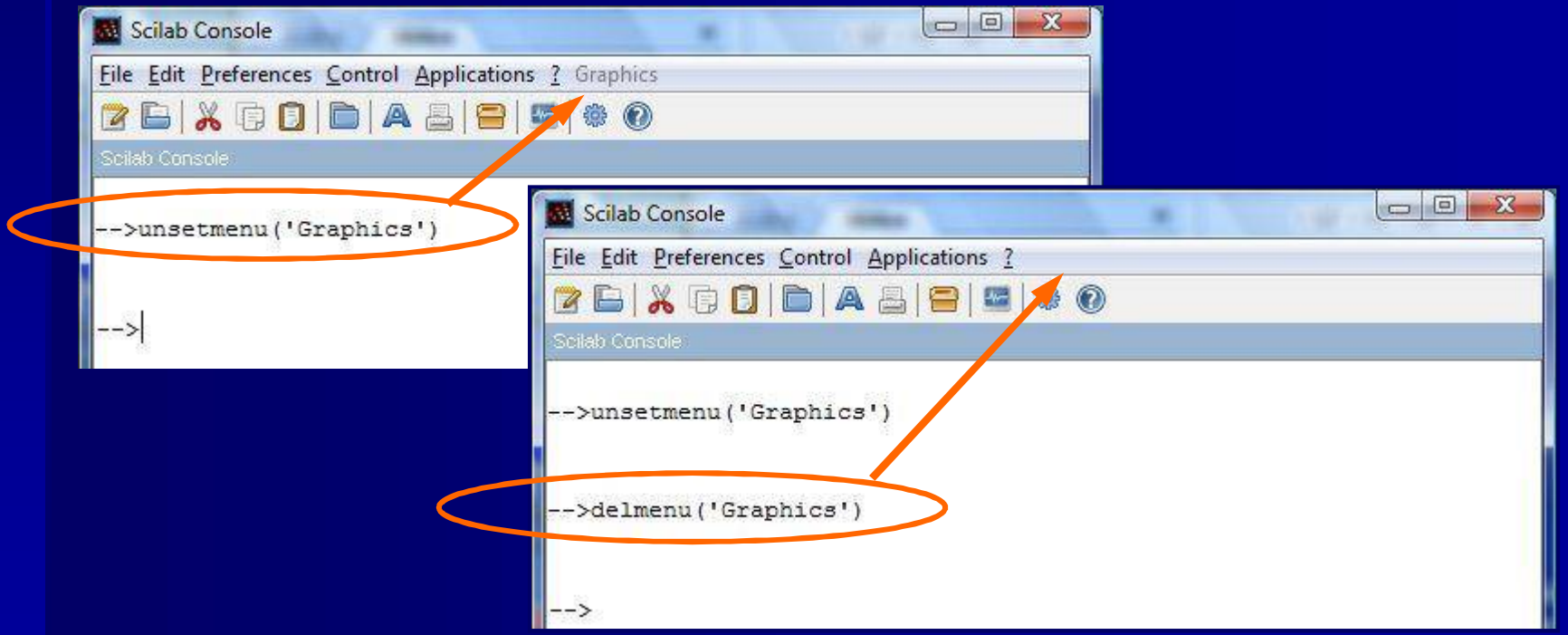
- Как демонстрация сказанного, здесь это команда, которая добавляет графическое меню, с подменю New Window и Clear
- Окно, в строке меню Консоли:

```
-->addmenu('Graphics',['New window','Clear
window','Close window'])

-->Graphics
  Graphics

!scf()  clf()
```



Прошив окна (3/3)

- Вы можете убедить себя, что добавленное меню Консоли работает нажав на кнопку "В новом окне", чтобы открыть графическое окно и нажмите "Закрыть окно", чтобы закрыть его снова
- Как следующим шагов мы можем отключить созданное меню, командой unsetmenu () и удалите его delmenu ():



Взаимодействие с Графическим окном (1/4)

- Scilab имеет многочисленные команды для взаимодействия с графическим окном; среди них являются:

xclick()	Ожидает щелчком мыши, возвращает значение) число окне, где нажмите происходит, б) положение мыши и в) номер клавишей мыши используется (левый, центральный, правый)
xgetmouse()	Возвращает текущее положение курсора мыши
seteventhandler()	Устанавливает обработчика событий для текущего графического окна
seteventhandler('')	Удаляет обработчика

Следующий скрипт составлена на основе Помощь / xgetmouse. Это рисует прямоугольник на окне с графикой. Прямоугольник начинается в месте расположения указателя мыши на первом щелчком левой клавишей, и замораживает прямоугольник вторым щелчком.

Взаимодействие с GW (2/4): сценарий (1/2)

Посмотрите на описания data_bounds под Справка / axes_properties (не очень полезно)

По словам Помощь / xclick первый вектор элемент должен быть числовым, но Scilab требует имя

Посмотрите на аргументы xrect (), это те, с которыми позже мы работаем

Третий элемент вектора равен -1, и курсор мыши переместился (см. функции Помощь обработчика / событие)

```
// rectangle_selection.sce

// The script demonstrates the use of the mouse-related       /
// commands xclick(), xgetmouse() and xrect() when they       /
// are used to draw a rectangle in the Graphics Window        /

clear,clc,clf;

// Initialize drawing process:
//------------------------------------
a = gca();                                  // Get current Axes
a.data_bounds = [0 0;100 100];  // Boundaries for x & y coordinates
xtitle('Click left mouse button & drag to create a rectangle. ...
  Click a second time to freeze')        // Display instruction
show_window();                              // Put Graphics Window on top

// Start drawing rectangle in the Graphics Window:
//-------------------------------------------------------------------
[button,x_coord,y_coord] = xclick(); // Point of mouse button click
xrect(x_coord,y_coord,0,0)
                  // Start rectangle at mouse pointer x & y coordinates
rectangle = gce();                          // Get rectangle handle
mouse = [x_coord,y_coord,-1];       // Mouse pointer 1x3 matrix
```

Взаимодействие с GW (3/4): сценарий (2/2)

Цикл начинается с проверки состояния мыши. Напомним, от предыдущего слай да вектор мышь = [x_coord, y_coord,-1]

После этого новые данные вычисляются для прямоугольника

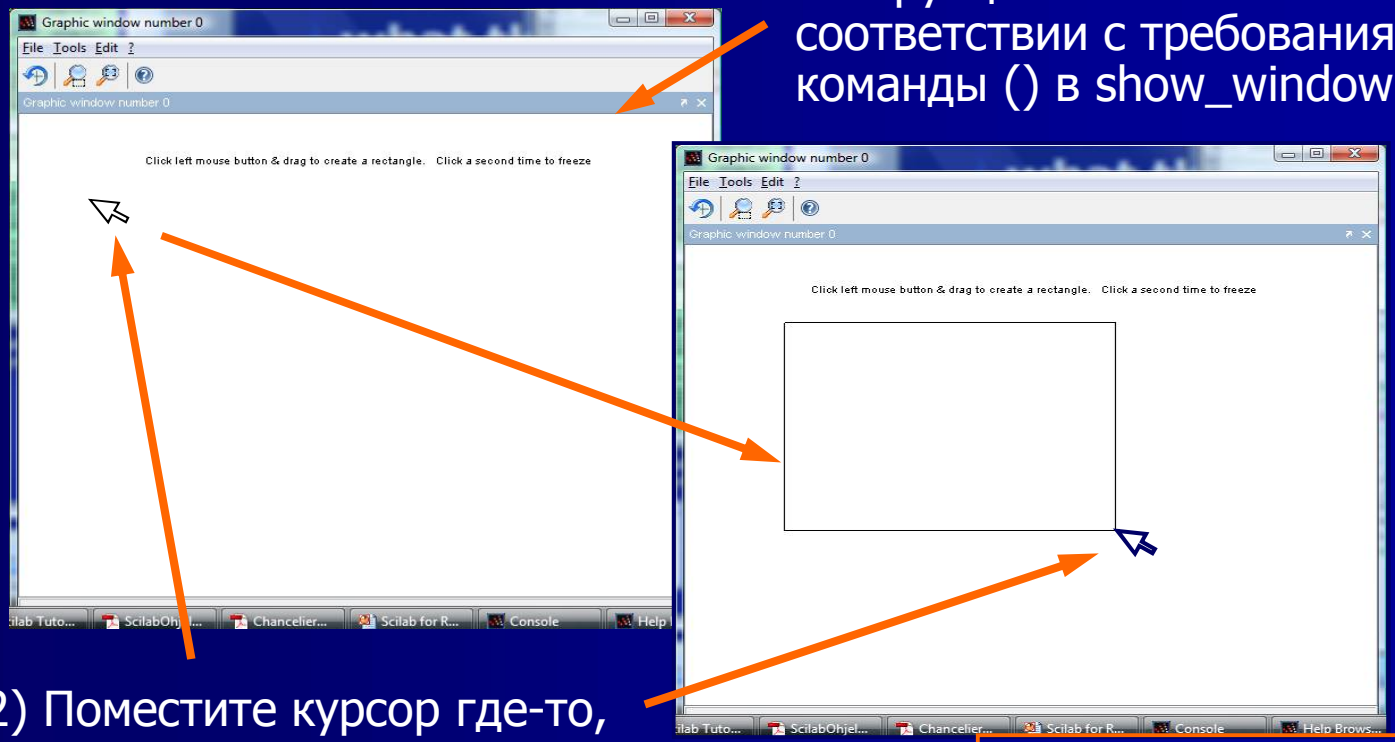
Последний штрих, чтобы определить новые значения ручки (см. xrect () аргументы выше)

```
// Execute mouse commands for rectangle:
//-------------------------------------------------------
while mouse(3) == -1 do                  // Repeat until second click
  mouse = xgetmouse();                   // Check mouse position
  x_coord1 = mouse(1);                   // Mouse location in x-plane
  y_coord1 = mouse(2);                   // Mouse location in y-plane
  x_origin = min(x_coord,x_coord1);      // Define x origin
  y_origin = max(y_coord,y_coord1);      // Define y origin
  width = abs(x_coord-x_coord1);         // Define width of rectangle
  height = abs(y_coord-y_coord1);        // Define height of rectangle
  rectangle.data = [x_origin,y_origin,width,height];
                 // Change rectangle origin, width and height
end
```

А-до-конца цикла работает вечно, если второй кнопки мыши Кликните изменяет состояние мышь (3) == -1. Если состояние ожидания быть добавлены в цикле.

Взаимодействие с GW (4/4): что он делает

1) В графическом окне с инструкцией всплывает в соответствии с требованиями команды () в show_window



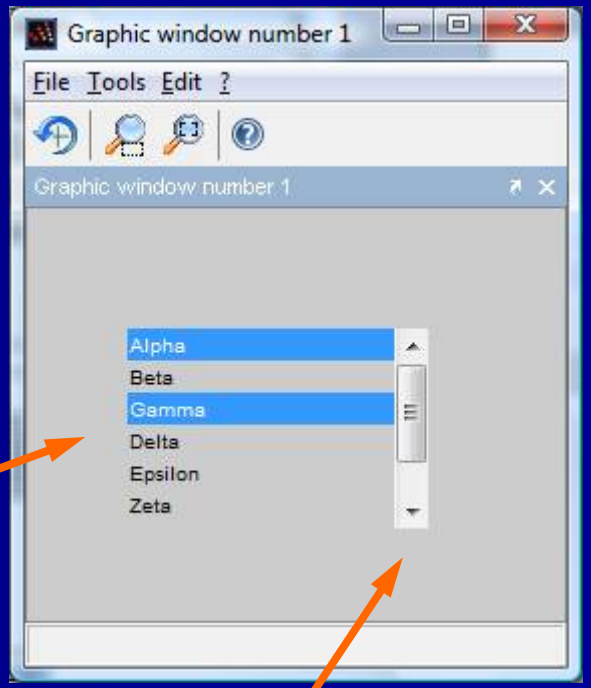
2) Поместите курсор где-то, нажмите и перетащите, и нажмите еще раз, чтобы заморозить

Что делать с этой функцией? Понятия не имею ....

: Introducing GUI демо 1: Введение figure() & uicontrol()

- Здесь figure() генерирует фигуру (открывает графическое окно), UIControl () создает объект графического интерфейса пользователя в GW, и два из пунктов в списке выделены set()
- Центральный аргумент в данном случае 'окно списка ", который определяет список

```
// uicontrol-1.sce /

// A basic GUI exercise /

clc; xdel();

f = figure();                 // Create a figure
h = uicontrol(f,'style','listbox',..          // Create a listbox,...
    'position',[50 300 150 100]);          //     h = handle
set(h,'string',"Alpha|Beta|Gamma..    // Fill the list
    |Delta|Epsilon|Zeta|Eta|Tau");
set(h,'value',[1 3]);    // Highlight items 1 and 3 in the list
```



Обратите внимание на полосу прокрутки, она всплывает, когда высота слишком мала для всех элементов

GUIs: всплывающие в окне функции

- Scilab имеет несколько команд для создания всплывающих окон. Обратите внимание, что x_message () является устаревшим и не будет работать в Scilab 5.2 и более поздних версиях; вместо этого должна быть использована MessageBox ():

команда	особенность
messagebox()	Презентация сообщения (см. демо 2, случаи 1, 2 и 7)
x_choose()	Альтернатива выбирается из списка (Demo 2, случай 3)
x_choices()	Как и предыдущий, но с несколькими вариантами (Демо2, Случай 5)
x_dialog()	Окно с диалоговым многострочным окном (Demo 2, случай 4)
x_mdialog()	Как и предыдущий, но с несколькими параметрами
x_matrix()	Окно ввода Вектора / матрицы (Демо 2, кслучай 6)
list()*	Создает список объектов (Demo 2, случай 5)

GUIs: messagebox()

Синтаксис MessageBox функции () состоит в следующем

```
MessageBox ("сообщение", "название", "икона", ["кнопки"], "модальный »)
```

Сообщение, что вы хотите передать

Ячейка в заголовке (по умолчанию это "Scilab Сообщение")

Икона должна быть помещена в поле

Вектор строк с кнопки легенд

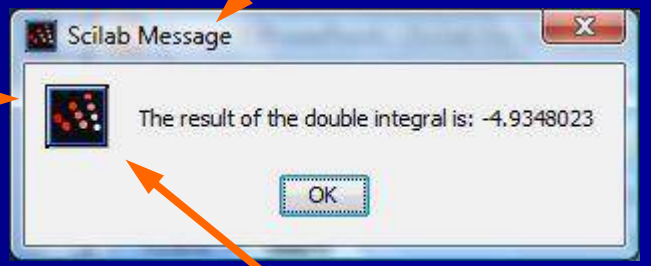
«Модальный» рассказывает Scilab ждать действий пользователя (иначе возвращается 0)

Определяемые иконки:
"Ошибка", "песочные часы", "Информация", "PASSWD", "Вопрос" и "предупреждение”

GUI :демо-2: создание всплывающих окон (1/5)

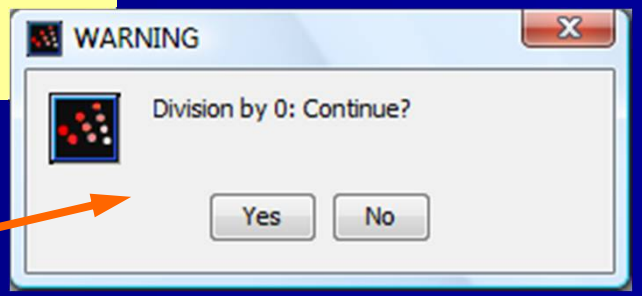
Случай 1:Напомним, что это всплывающее окно было создано путем добавления команды MessageBox = ('Результат двойной интеграл:' + строка (Я)) в конце сценария примера 5-4

Стандартное название



иконка по умолчанию

```
-->m = messagebox('Division by 0: Continue?','WARNING',['Yes' 'No'])

 m  =
```

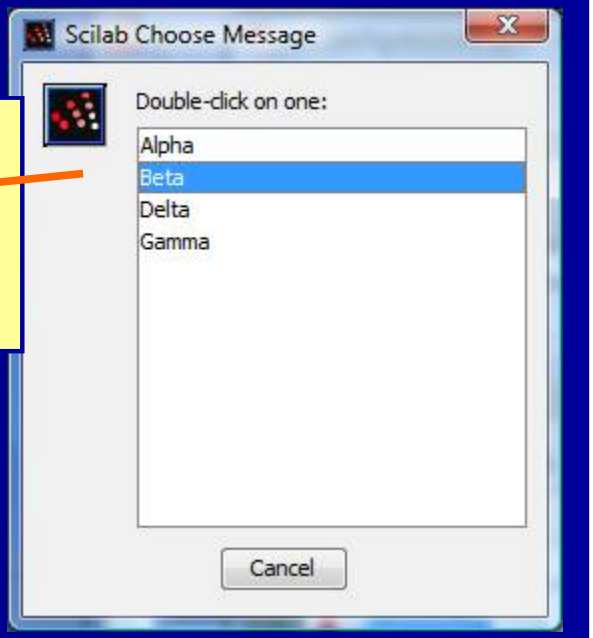


Случай 2 не так! Да / Нет кнопки не имеют никакого значения, так как в случае не объявлен "модальный" и Scilab дизайн возвращает нулевое значение по умолчанию (0)

GUI demo 2: создание всплывающих окон (2/5)

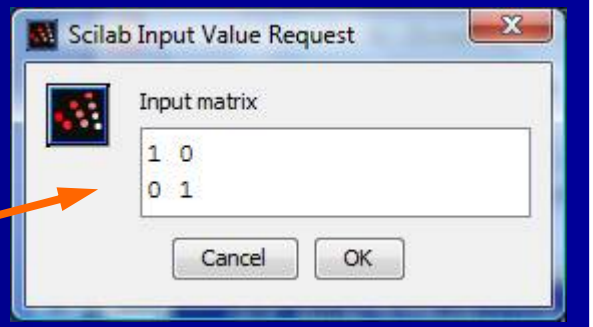
Случай 3: x_choose () с четырьмя альтернативами

```
-->ans=x_choose(['Alpha','Beta','Delta','Gamma'],'Double-click on one:')

 ans  =
```



Случай 4 x_dialog () с участием превращается строки в матрицу

```
-->answer=evstr(x_dialog('Input matrix',['1 0';'0 1']))
 answer  =
   9.    8.    7.
   1.    2.    3.
   6.    5.    4.
```



Изменение матрицы по мере необходимости, нажмите кнопку ОК

GUI demo 2:создания всплывающих окон (3/5)

```
// x-choices_demo.sce

// Demonstrates the x_choices() command   /

clear,clc;

list1 = list('Food grading',3,['Excellent','Good','Passable','Poor']);
list2 = list('Service grading',2,['Excellent','Good','Passable','Poor']);
list3 = list('Interior grading',4,['Excellent','Good','Passable','Poor']);
answer = x_choices('Grade restaurant food..
       service & interior',list(list1,list2,list3))
```

Случай 5: x_choices () с четырьмя альтернатив для трех случаев


```
-->answer
 answer  =


   3.    2.    1.
```

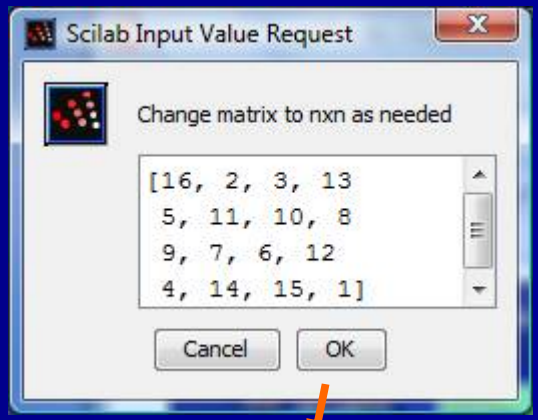
Возьмите свой выбор, нажмите кнопку ОК, и Scilab возвращает ответ в виде вектора *

*) Scilab 5.1.1 возвращает ответ автоматически, с 5.3.1 и 5.3.2 следует запрос (ошибка?)

GUI demo 2:создания всплывающих окон (4/5)

Случай 6: Вычислить определитель для матрицы A, заданного x матрицы (). Предполагается, матрица 3x3 идентичность

```
// x-matrix_demo.sce

// Demonstrates the use of x_matrix()    /

clear,clc;

A = x_matrix('Change matrix to nxn as needed',eye(3,3));
det(A)                  // Calculate determinant
clean(det(A))           // Clean det(A) for small values
```



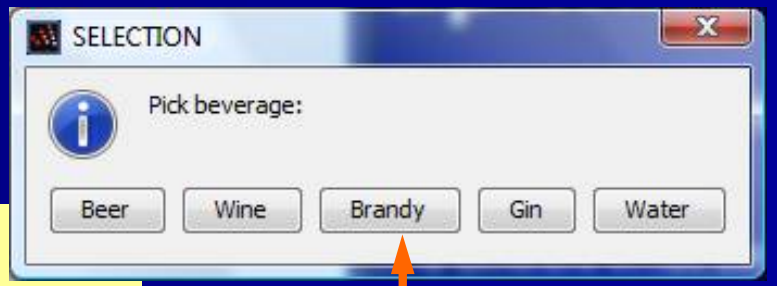
Изменение (здесь, чтобы 4x4 Магический квадрат) и нажмите кнопку ОК

```
ans  =


 - 1.450D-12

ans  =
```

Ответ такой же, как раньше, в главе 5 Проблема:. Он работает в Scilab 5.1.1, но не в 5.3.1 и 5.3.2

GUI demo 2:создания всплывающих окон (5/5)

Случай 7: Создание список вариантов напитков с использованием list() и окна messagebox()




```
-->bew = ['Beer','Wine','Brandy','Gin','Water'];

-->m = messagebox('Pick beverage','SELECTION','info',bew,'modal');
-->m
 m  =
   3.
```

Здесь выбран" Brandy " и ответ возвращается *

```
-->r = messagebox('Pick','Title',' ',['1','2'],'modal')

 r  =
   2.
```

*) Те же проблемы повторяется. Scilab не возвращает ответ автоматически (в Scilab 5.3.1 она сделала это с аналогичной случае, но не в 5.3.2 больше)

GUI:размер экрана компьютера и глубина цвета

- Размер экрана компьютера необходим, если мы хотим позиционировать графический интерфейс в определенной позиции в поле зрения
- Для этого нам нужна информация о размере экрана компьютера. Это может быть извлечено аргументом screensize_xx. Есть несколько альтернатив для суффикса _xx, проверьте Помощь / root_properties
- Другой альтернативой является количество бит разрешения дисплея цвета. Он может быть найден с аргументом screendepth
  Эти аргументы используются с функцией Get (), что означает "узнать". См. Пример 6-1 практического случае

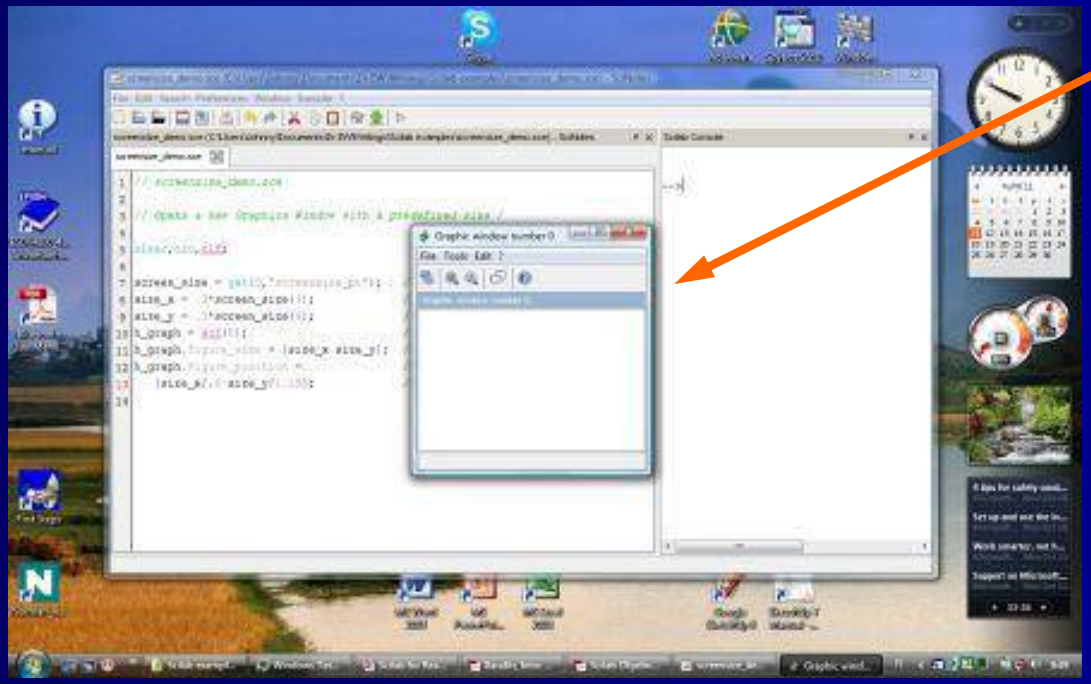
```
-->get(0,"screensize_px")
 ans  =

    1.    1.    1280.    800.

-->get(0,"screensize_pt")
 ans  =

    0.    0.    960.    600.

-->get(0,"screensize_norm")
 ans  =

    0.    0.    1.    1.

-->get(0,"screendepth")
 ans  =

    24.
```

GUI demo 3:открытие предопределенный GW, сценарий

- Эта демонстрация показывает, как открыть новое графическое окно с заданным размером и положением
- Размер определяется по отношению к размеру экрана компьютера в точках
- Положение в центре экрана должно быть найдено путем проб и ошибок

```
// screensize_demo.sce

// Opens a new Graphics Window with a predefined size & location /

clear,clc,clf;

screen_size = get(0,"screensize_pt");    // Find computer screen size
size_x = .3*screen_size(3);              // .6*screensize_pt 3rd element
size_y = .5*screen_size(4);              // .8*screensize_pt 4th element
h_graph = scf(0);                        // Open Graphics Window
h_graph.figure_size = [size_x size_y];   // Define GW size
h_graph.figure_position =...             // Position GW in the...
   [size_x/.6 size_y/1.15];              // middle of the screen
```

GUI demo 3:предопределенный GW, Скриншот



Небольшой GW открывается в середине экрана (картина была сжата и выглядит запутанной)

Заметим, однако, что размер GW не совсем пропорционально определенном соотношении от размера экрана, и это также меняется, если выбрать screensize_px вместо screensize_pt

GUI недостатки

- GUIs не совершенствуется в Scilab.(Грязный) текст на графических интерфейсах в WIKI.Scilab.org / HOWTO / говорит об очень старых ошибках, которые остаются нерешенными
  - Помимо того, что упомянуто в Demo 2, Случаи 5-7, и в конце концов обсуждения Примерами 6-1, я пережил проблемы с
  - Демо 1, где ListBox может (или не может) течь по оконной раме
  - Пример6-1, где этикетки слайдера и первой радиокнопки иногда открываются с уменьшенным размером шрифта
  - WIKI.Scilab.org / HOWTO / также упоминает следующие ограничения:
  - Scilab не позволяет вертикальных слайдеров
  - checkbox == radiobutton
  - слайдер не имеет smallstep, никаких побочных стрелки (и как я узнал с примера 6-1, Scilab получает блокировку, когда я перетаскиваю ползунок)
  - цвет переднего плана всегда серый
  - Pressed радио / проверка всегда бледно-красного (не пробовал)

Dr. EW
Johnny Heikell

16. Обработка файла

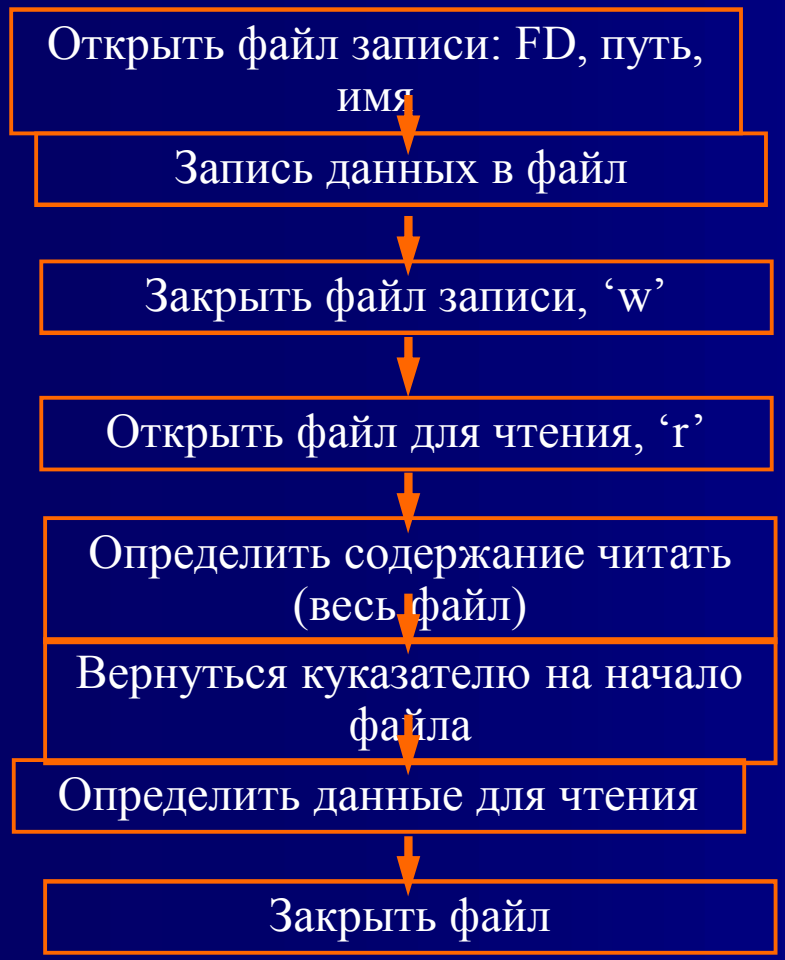
Мы должны обработать
например файл для обработки
данных измерения.

Обработка файла: введение

- В машиностроении, данные о внешних файлах часто происходят в автоматизированных измерениях. Данные должны быть прочитаны Scilab, прежде чем он сможет их обработать. Мы сосредоточим наше обсуждение на этом аспекте работы с файлами,
Scilab имеет набор команд для работы с файлами, начиная с команды mopen (), которая открывает файл, и mclose (), которая закрывает его. Между этими двумя командами мы используем например такие, как:

mfprint, fprintMat()	Запись данных в файл (fprintMat () для матрицы файлов)
mfscanf(), fscanMat()	Чтение файла (fscanMat () для матрицы файлов)
mseek()	Перемещение курсора
menf()	Проверьте конец файла
size()	Проверьте размер объекта

Обработка файла: демо 1 (1/5), введение



В этой демо-версии Scilab создает файл данных для чтения;
Позже, для себя, мы узнаем, как читать эти текстовые файлы, созданные другими программами

Последовательности сценариев показаны с правой стороны. Сценарий демонстрирует использование функций mopen (), mclose (), mfprintf (), mseek (), и mfscanf ()
Обратите внимание на следующие действия:

Откройте как 'W' файл, недалеко 'W' файлу, открытый в виде файла 'R', недалеко 'R' файла. Указатель стека перемещается вниз, по мере того, как мы пишем файл, но он должен быть возвращен в начало прежде, чем мы начнем читать.

Обработка файла: демо 1 (2/5), сценарий

Создание текста (. TXT) файл с mopen (). FD = дескриптор файла. Обратите внимание на аргумент 'W' ("запись"), который используется для создания нового файла

Затем заполните файл с данными (в данном случае создан т), используя mfprintf (). Обратите внимание на странный аргумент?% 6.3f \ п ', который определяет размер вывода (см. ниже)

```
// file_exercise1.sce

// The script demonstrates the process of 1) creating a text file    /
// on Scilab, 2) closing it, 3) opening it again to be written into,  /
// 4) writing the actual data into the file, 5) reading certain       /
// pieces of data from the file, and 6) closing the read file,        /
// Notice that both close operations are necessary!                   /

clear,clc;

// Create and open a text file for the exercise:
//-----------------------------------------------------------
fd = mopen('H:\Dr.EW\Writings\Scilab examples\file_exercise1.txt','w');

// Create data and write into the exercise file:
//-----------------------------------------------------------
t = (1:1:18)';                          // Integers from 1 to 18
mfprintf(fd,'%6.3f\n',t);
```

Обработка файла: демо 1 (3/5), сценарий продолжение

После этого файл должен быть закрыт

Тогда снова открылась для чтения («r»)

Далее мы читаем в полном объеме (в -1)

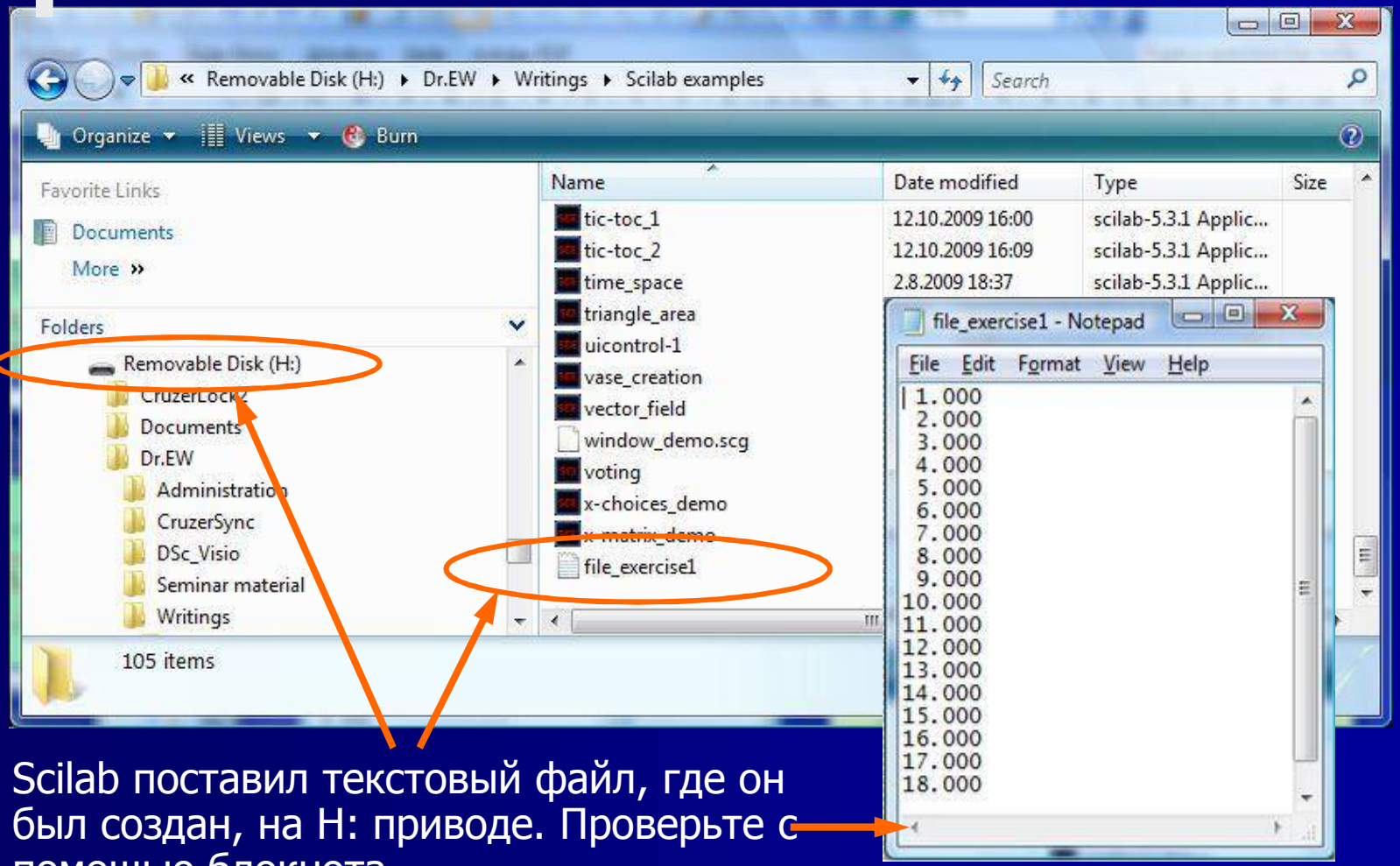
Но указатель должен быть возвращен в начало ..

прежде чем мы сможем определить, какие данные мы хотим видеть

Закончить, закрыв файлу (см. ниже примечание на mclose ()

```
// Close exercise file:
//-------------------------
mclose(fd);

// Open the exercise file for reading:
//-------------------------------------------
fd = mopen('H:\Dr.EW\Writings\Scilab examples\file_exercise1.txt','r');

// Read and format file contents:
//-----------------------------------------
contents = mfscanf(-1,fd,'%f')       // -1 means entire file contents

// Return position pointer to file beginning:
//-------------------------------------------------
mseek(0,fd)   // Following mfscanf(-1, , ) the pointer is at the end

// Read some data from the file:
//---------------------------------------
five_data = mfscanf(fd,'%f %f %f %f %f')       // First five data
three_data = mfscanf(fd, '%f %f %f')           // Next three data
[n,data_9,data_10,data_11] = mfscanf(fd,'%f %f %f')// Three specific..
                                                //    elements
// Close the file:
//-----------------
mclose(fd)
```

Обработка файла: демо 1 (4/5), текстовый файл.



Scilab поставил текстовый файл, где он был создан, на H: приводе. Проверьте с помощью блокнота

Обработка файла: демо 1 (5/5), проверка

Определенное для чтения содержимое переменной воспроизводит содержимое текстового файла на консоли

Затем мы можем выделить конкретные элементы из списка

```
-->contents

 contents  =

  1.
  2.
  3.
  4.
  5.
  6.
  7.
  8.
  9.
  10.
  11.
  12.
  13.
  14.
  15.
  16.
  17.
  18.
```

```
-->five_data

 five_data  =

    1.    2.    3.    4.    5.

-->three_data

 three_data  =
    6.    7.    8.
-->data_11

 data_11  =

     11.
-->n
 n  =
   3.
-->data_9:11
 ans  =
    9.    10.    11.
```

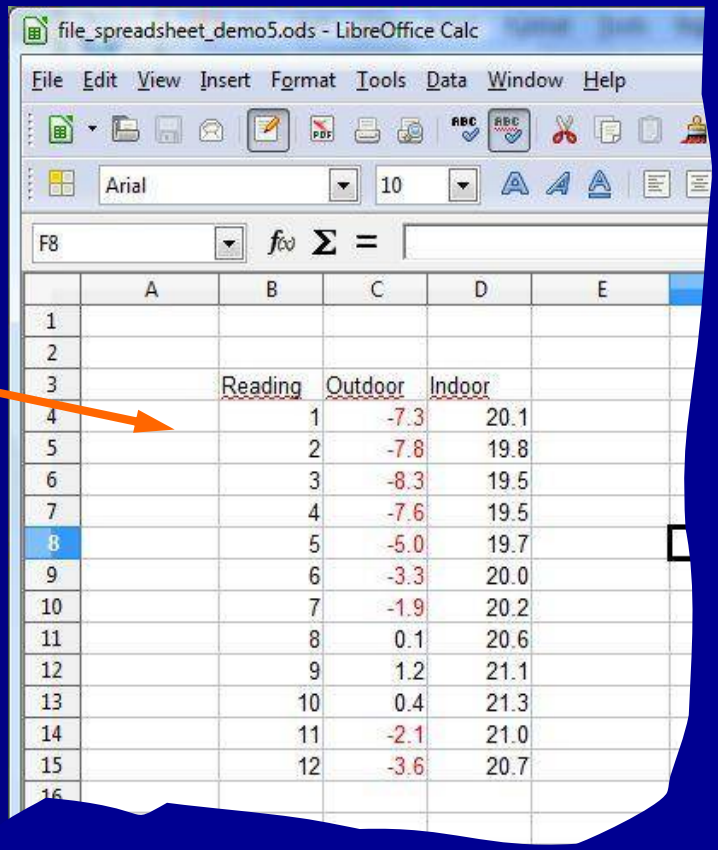
Переменные five_data, three_data и data_11 были определены в сценарии

n # элементов (-1) в векторе это принадлежит (4-1)

Мы также можем решать конкретные элементы в вектор-столбец и получить ответ в виде вектора-строки

Данные электронной таблицы (1/7): Создание данных

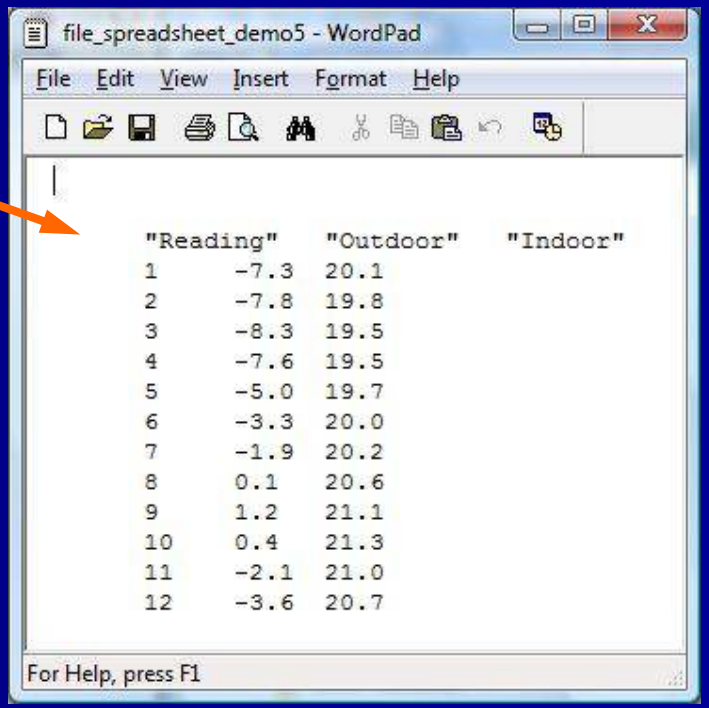
- Scilab не взаимодействует непосредственно с электронными таблицами. Данные должны быть сохранены в виде текстового файла?
- Я начал с новой страницы в блоке, LibreOffice Calc. Данные внутреннего/внешнего измерения температуры
- Процесс сохранения LibO и OOo данных в виде текстового файла. CSV объясняется позже
- Если вы делаете это в Excel, вы просто сохраняете его как текст (с разделителями табуляции). Не выбирайте текст Unicode, потому что Scilab не может прочитать его



	A	B	C	D	E
1					
2					
3		Reading	Outdoor	Indoor	
4		1	-7.3	20.1	
5		2	-7.8	19.8	
6		3	-8.3	19.5	
7		4	-7.6	19.5	
8		5	-5.0	19.7	
9		6	-3.3	20.0	
10		7	-1.9	20.2	
11		8	0.1	20.6	
12		9	1.2	21.1	
13		10	0.4	21.3	
14		11	-2.1	21.0	
15		12	-3.6	20.7	

Данные электронной таблицы (2/7): Данные, сохраненные в формате CSV.

- А вот данные LibO сохраняются как file_spreadsheet_demo5.csv в WordPad (рис. 5 отражает тот факт, что моя пятая попытка получилась)
- Давайте посмотрим, если Scilab может прочитать файл формата. CSV. Есть два варианта команды:
  M = fscanfMat () для матрицы вещественных чисел (текстовые данные игнорируются)
  [M, текст] = fscanfMat () для струнного матрицы
- Выход для обоих вариантов показан на следующем слайде
  После этого мы можем написать сценарий для построения данных



Данные электронной таблицы (3/7):. Файл CSV читать Scilab

```
-->M = fscanfMat('I:\file_spreadsheet_demo5.csv')
 M  =

    1.      - 7.3       20.1
    2.      - 7.8       19.8
    3.      - 8.3       19.5
    4.      - 7.6       19.5
    5.      - 5.        19.7
    6.      - 3.3       20.
    7.      - 1.9       20.2
    8.        0.1       20.6
    9.        1.2       21.1
    10.       0.4       21.3
    11.     - 2.1       21.
    12.     - 3.6       20.7
```

M = fscanfMat()

[G,text] = fscanfMat()

```
-->[G,text] = fscanfMat('I:\file_spreadsheet_demo5.csv')
 text  =

            “Reading”       “Outdoor”       “Indoor”
 G  =

      1.      - 7.3       20.1
      2.      - 7.8       19.8
      3.      - 8.3       19.5
      4.      - 7.6       19.5
      5.      - 5.        19.7
      6.      - 3.3       20.
      7.      - 1.9       20.2
      8.        0.1       20.6
      9.        1.2       21.1
      10.       0.4       21.3
      11.     - 2.1       21.
      12.     - 3.6       20.7
```

Примечание: Если вы работаете с MS Excel вы используете конечно концовку текстовый вместо CSV (CSV расшифровывается как разделенные запятыми Переменная)..

Данные электронной таблицы (4/7): Скрипт для построения (1/2)

Команда fscanfMat () не может быть разделена на два ряда (даже если это не требуется в данном случае)

Размер (название, 'R') функции используется для определения количества строк матрицы

Столбцы матрицы образуют отдельные векторы

```
// spreadsheet_data_plot.sce

// The script reads data from the test file          /
// file_spreadsheet_demo5.csv, determines its    /
// length,and plots its two measurement sets     /

clear,clc,clf;

// Open the file, determine number of rows,
// and form vectors of its columns:
//---------------------------------------
data_file = fscanfMat(IH:\file_spreadsheet_demo5.csv');
                    // Opens text file
rows = size(data_file,'r');      // Determine number of rows
readings = data_file(:,1);       // Column 1, reading # (redundant)
outdoor = data_file(:,2);        // Column 2, outdoor temperature
indoor = data_file(:,3);         // Column 3, indoor temperature
```

Данные электронной таблицы (5/7): Скрипт для построения (2/2)

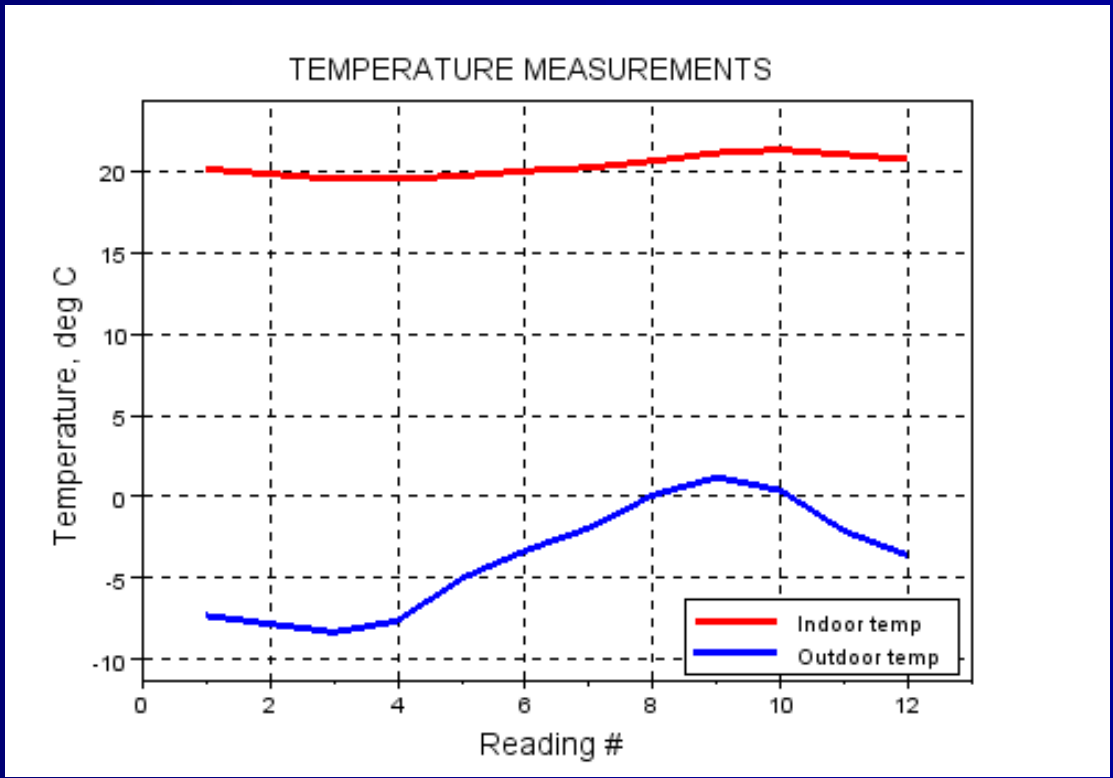
Команда использует устаревшие участок plot2d () синтаксис, который мы видели раньше. Причина, по которой он здесь то, что plot2d () с frameflag аргумента новый синтаксис не работает, когда два графика должны быть слиты в одном участке:
Второй участок уничтожает первый, и, когда аргумент прямоугольника входит, Scilab отвечает сообщением об ошибке (это пробовал, знаю)

```
// Assume outdoor temp always lower
// than indoor and form plot commands:
//-----------------------------------------
ymin = min(outdoor);                  // Determine min temp
ymax = max(indoor);                   // Determine max temp
dy = (ymax-ymin)/10;                  // Define frame margin
rect = [0,ymin-dy,rows+1,ymax+dy];    // Define plot frame
x = linspace(1,rows,rows);
plot2d(x,indoor,5,'011',' ',rect)     // Plot inodr temp
plot2d(x,outdoor,2,'000')             // Plot outdor temp
  xgrid(1)                            // Add grid
  xtitle('TEMPERATURE MEASUREMENTS','Reading #',...
         'Temperature, deg C')
  legend('Indoor temp','Outdoor temp',4)
```

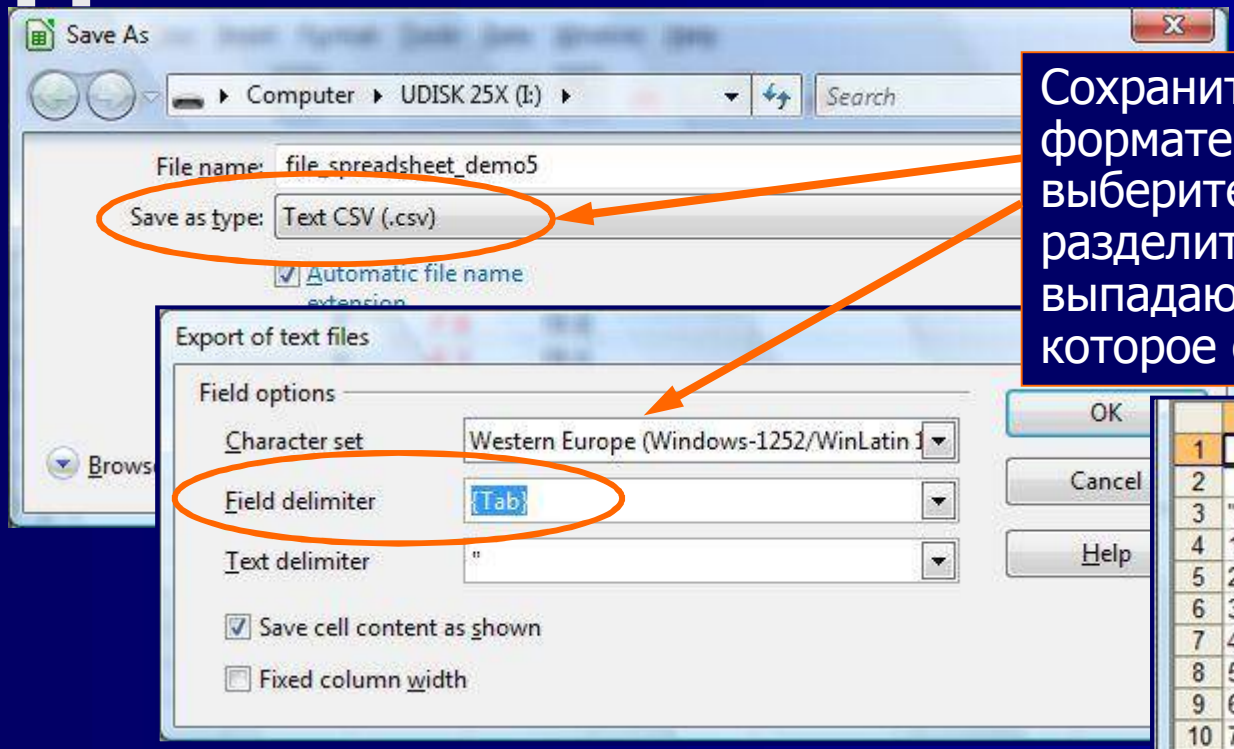
Данные электронной таблицы (6/7): Участок

Простой сюжет, но суть в этом упражнении заключается в том, чтобы показать, как перейти от данных электронных таблиц в текстовый файл, а затем для построить график данных

И тогда мы обратимся к вопросу о том, как создавать текстовые файлы с LibreOffice Calc и OpenOffice.org Calc (следующий слайд)



Данные электронной таблицы (7/7): Текст данные в LIBO & OOo



Сохранить как текстовом формате CSV (. CSV) и выберите вкладку в разделитель полей выпадающего меню окна, которое открывается.

	A	B	C
1			
2			
3	"Reading""Outdoor""Indoor"		
4	1-7.320.1		
5	2-7.819.8		
6	3-8.319.5		
7	4-7.619.5		
8	5-5.019.7		
9	6-3.320.0		
10	7-1.920.2		
11	80.120.6		
12	91.221.1		
13	100.421.3		
14	11-2.121.0		
15	12-3.620.7		
16			
17			

Сохраненный. Файл CSV выглядит некрасивым, если вы открываете его с Excel, но хорошим, если открыть его в блокноте и WordPad

mopen()

- Функция mopen () является, конечно, сложнее, чем то, что можно понять из вышеизложенного. Забыв двоичные и текстовые файлы, общей структурой mopen () является:[FD <, ошибаться>] = mopen (имя_файла <, режим>), где имя_файла является всем путем к файлу, включая его имя
- Режим определяет, что делать с данными, например:
  r, для чтения существующего файла
  w,для создания нового файла и записей в него, alt. перезаписи данных в существующий файл
- a,добавьте, откройте файл и добавьте данные в конец
  fd, дескриптор файла, временное имя файла
  err, ошибаться, параметр ошибки. err = 0, если файл успешно открыт, err <> 0, если открытие файла не удалось (merror ()-функция, связанная с аргументом err)
- Это хорошая идея, чтобы проверить параметр err, после открытия файл (не было сделано в Demo 1)

mclose()

Файл, который был открыт с mopen () должен быть закрыт с помощью команды mclose (FD), даже если оно автоматически закрывается. Однако, обратите внимание на следующее двусмысленное заявление в Scilab Помощь Браузер:

"mclose необходимо использовать, чтобы закрыть файл, открытый mopen. Если FD опущен mclose закрывает последний открытый Файл.

Будьте осторожны с использованием [mclose ('все')] ... потому что, когда он используется внутри файла сценария Scilab, а также завершаются сценарий и Scilab не будет выполнять команды, написанные после mclose ('все') "

mfprintf(), fprintfMat()

- Команда mfprintf () используется для преобразования формата, и записи данных в открытом текстовом файле
- Общая структура mfprintf () является:? mfprintf (FD, "<text a> format_1 <text b> format_2? <text c> format_3 ... ', значение_1, значение_2, value_3 ...)
- Это означает, что каждое значение, которое мы хотим напечатать объявляется дополнительным текстом, форматы должны быть напечатаны (как внутри одной пары кавычек),для печати
- Декларации формата приведены на следующем слайде Формат демо, две горки вниз должны дать лучше понять что все это значит. Если вы спросите меня, это выглядит действительно грязно ...
- Команда fprintfMat () используется для записи матрицы в файл. Подробности см. в справке

Определение формата

- Напомним, для аргументов% 6.3f \ п 'и% F в обработке файлов Demo 1 Они являются частью набора определений формата.:
- % d для целых чисел (например 1230)
- % f для десятичных (например, 12,30987)
- % e для экспонент (например 1.2345e +002)
  % s для текста (строка) презентации (например Hello World!)
- % 6.3f определить размер выходного
  6 предназначен для общего количества фигур
  3 для количества цифр после запятой
- \ n "идут в новой линии"
  \ t "использовать горизонтальный табулятор"
  Некоторые комбинации могут быть определены, как% 6.3f \ п,

Формат демо: сценарий (1/2)

Это демо направлено на уточнение использование деклараций формата:

```
// file_format_demo.sce

// Demonstrates the use of mfprintf() format definitions.     /
// Pay attention that with several variable to be printed,    /
// all formats are declared (inside a single pair of citation /
// marks) before the variables are defined.                   /

clear,clc;

// Create a new test file for writing:
//------------------------------------
fd = mopen('H:\Dr.EW\Writings\Scilab examples\file_format_demo.txt','w');

// Some variable to play with:
//----------------------------
A = 123.45678901;
a = 0.3;
b = 1.23e-02;
c = a + %i*b;
text = 'Hello World';
```

Первоначальные заявления здесь.
Реальный материал на следующем слайде

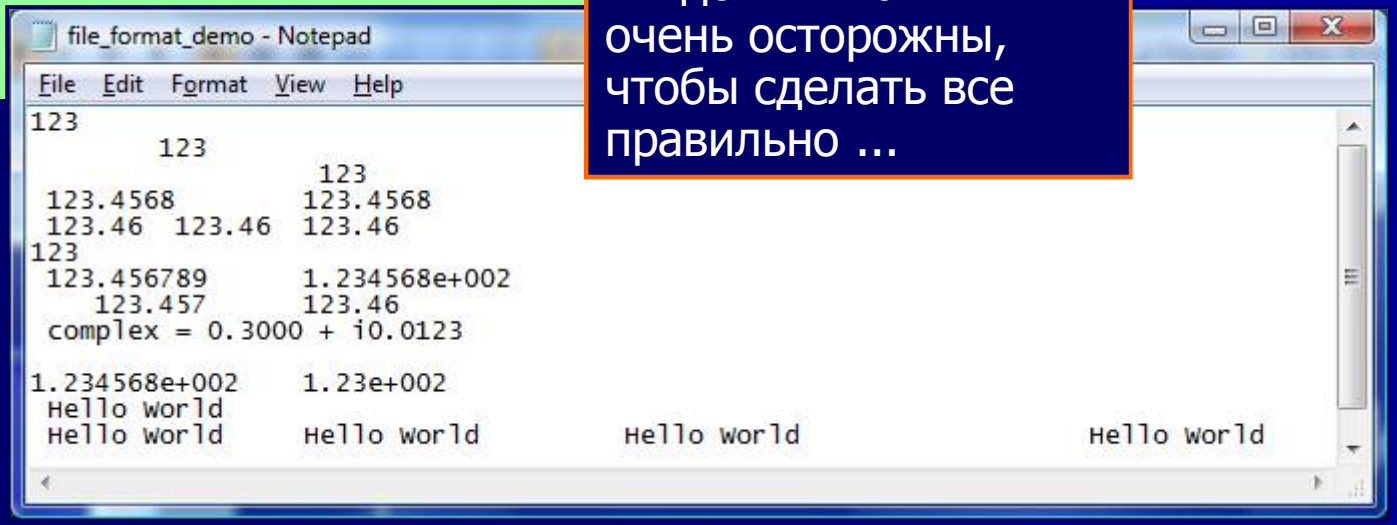
Формат демо: скрипт (2/2) и текстовый файл

```
// Several outputs to be demonstrated:
//-----------------------------------------
mfprintf(fd,'%d\n %10d\n %20d\n %8.4f\t %8.4f\n %5.2f\t %5.2f\t %5.2f\n',...
         A,A,A,A,A,A,A,A,A);
mfprintf(fd,'%d\n %f\t %e\n %10.3f\t %6.2f\n complex = %3.4f + i%3.4f\n\n',...
         A,A,A,A,A, real(c), imag(c));
mfprintf(fd,'%e\t %5.2e\n %s\n %5s\t %10s\t %15s\t %20s\t\n',...
         A,A, text, text, text, text, text);

// Close the opened file:
//-------------------------
mclose(fd);
```

необязательный текст не используется ни в одном из случаев

Не забудьте закрыть!

Вы должны быть очень осторожны, чтобы сделать все правильно ...



mfscanf(), fscanfMat()

- Мы использовали mfscanf () в Demo 1 читать (сканирование) данные из файла. Два примера его использования:
- содержание = mfscanf (-1, FD, "% e '). С помощью этого аргумента он читает все содержимое файла и форматирует его
- four_values = mscanf (FD, "% e% e% e% e '). Считывает четыре первых данных в файле
- После прочтения данных, указатель стека остается там, где она есть, и мы должны использовать mseek (п, е) команды, чтобы перевести его на новое место. Первая строка в стеке имеет номер 0, как указано mseek (0, FD) в Demo 1
- При обсуждении таблицы данных мы использовали функцию fscanfMat () для чтения данных, содержащихся в. CSV-файла. Функция имеет две альтернативных последовательности вызовов:
- fscanMat (файловый путь, <opt_arg>) читать числовую часть только скалярных данных матрицы в текстовом файле
- [M, текст] = fscanfMat (файловый путь, <opt_arg>) для чтения данных,которые включают первые нецифровые линии
  По умолчанию опция аргумент% 1 г. Проверьте с Помощь других вариантов

Dr. EW
Johnny Heikell

17. Анимация

Краткое введение в создание динамической графики

Введение.

- Анимация последовательности участков на окне с графикой; выполняется, показав сюжет, замораживание его во время дополнительных сдвигов в процессе расчета, а затем обмен старого сюжет на новый. * С правильной скоростью и шагом он дает иллюзию непрерывного движения
  Есть два основных режима для создания анимации:
  Реальный режим времени. Анимация работает в то время как скрипт выполняется, с скоростью , которая определяется пошаговыми сдвигами и быстродействием компьютера. Казнь может влиять (замедлять) по realtimeinit () и реальном времени () функций
  Режим воспроизведения. Возможно в Matlab с GetFrame и кино команд, но Scilab не поддерживает эту альтернативу
  Инструментом для создания анимации является команда растровое изображение, ручка и функция show_pixmap ().
  Пример 6-2, однако, не надо использовать команду растровое изображение

*) Если человек не хочет сохранить всю последовательность, как в примере 6-2.

Demo 1 (1/4): Introducing pixmap & xfarcs()

- Это демо составлено на основе Антонелли и Chiaverini. В нем представлены, в частности, растровое изображение и show_pixmap () пару команд
  растровое изображение = "на" / "выключено«
- Режим растровое изображение * используется для достижения гладкой анимации. С помощью команды растровое изображение ручки = "на" дисплей обновляется только тогда, когда обращаешься к команде show_pixmap ()
- Сравните этот случай с drawlater () - drawnow () пар в обычном черчении
- Сценарий использует xfarcs функции (), чтобы заполнить движущийся пирог. См. также функции Scilab являются xfarc (), xarcs (), и Xarc () xfarcs () используется вместо xfarc (), поскольку последняя не предусматривает для определения участка цвет аргумента, а его ось ручка GCA () не признает никаких детей, которые позволили бы определить цвет.

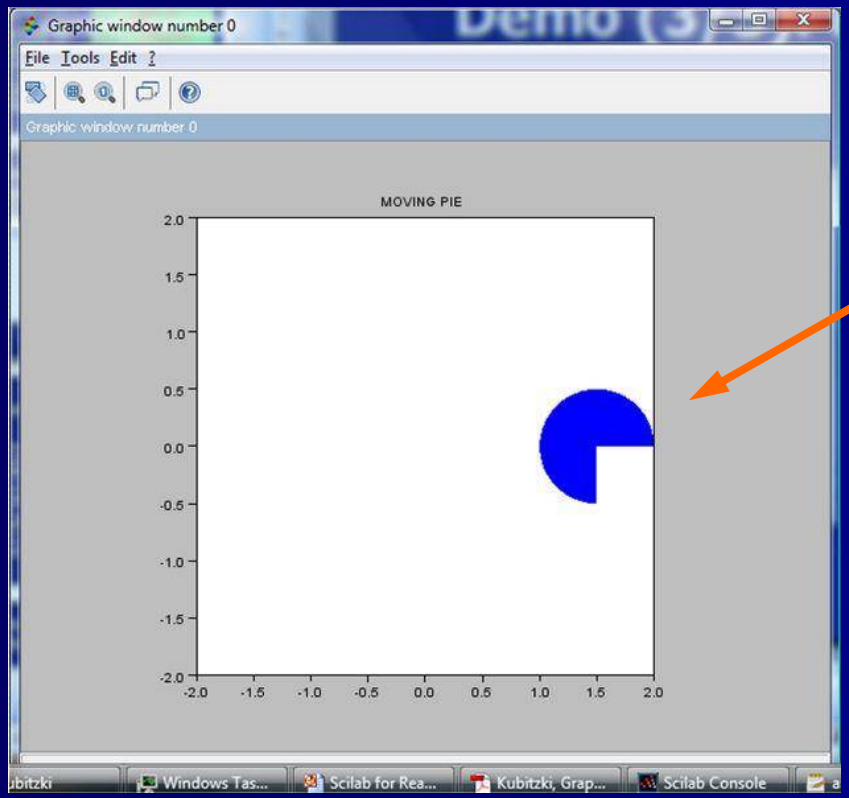
*) Также называется "режим двойной буфер", потому что картина создается впервые в одном буфер перед толкают на секунду (Графика Window).

Demo 1 (2/4): moving pie, script

- Отсутствующие x и y значения замещены (% NaN)
- Только frameflag = 3 работает в этом случае Обратите внимание на мнимые значения theta1 & theta2. Их относительные значения (2п & 10п) определяют пять петель, которые делает пирог до окончания полного круга xfarcs () требуется шесть значения вектора в качестве аргумента.
- Цветовой код не является обязательным (цвет значения по умолчанию черный)

```
// animation_ball.sce

// Creates a cut pie that makes 5 loops while          /
// moving around a circle. Demonstrates the use of     /
// the pixmap - show_pixmap() pair of commands,        /
// and the use of xfarcs() in drawing                  /

clear,clc;

steps = 250;     // # of animation steps
r1 = 0.5;        // Pie size
r2 = 0.5;        // Loop size
f = gcf();       // Figure handle
f.pixmap = "on";       // Create before display
for i=1:steps
  clf();      // Erase pie after each step
  plot2d (%nan,%nan,frameflag=3,..  // Define figure
    rect=[-2,-2,2,2],axesflag=1)
  xtitle('MOVING PIE');
  theta1 = i*2*%pi/steps;
  theta2 = i*10*%pi/steps;
  c = [cos(theta1)+r2*cos(theta2),..   // Define pie..
    sin(theta1)+r2*sin(theta2)];       //    position
  xfarcs([c(1)-r1, c(2)+r1, 2*r1,...   // Plot pie,..
    2*r1, 0, 360*48]', 2);             //    color=2
  f.background = color('grey');
  show_pixmap();      // Display created graphics
end
f.pixmap = 'off';       // Exit pixmap mode
```

Демо 1 (3/4): перемещение кругового, замороженного участка



Вот синий сектор в сочетании стартовой и финишной позиции

Завершение полного круга в 250 шагов занимает около 10 секунд с моим 1,6 GHz двухъядерным процессором

Demo 1 (4/4):обсуждение

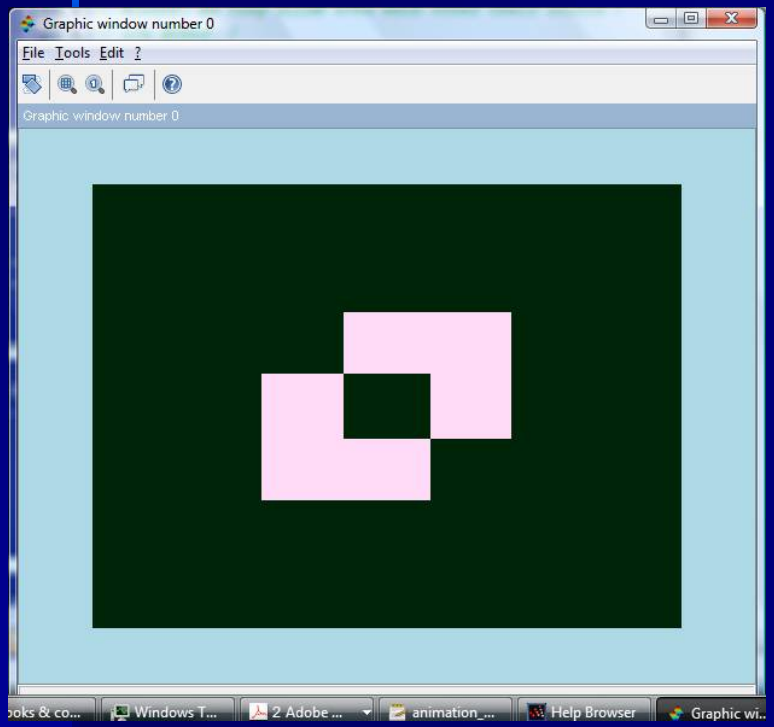
- Происходило что-то странное в то время, когда я пытался получить это
- Окно графика в основном открыто, как показано выше, но я также видел черный шар (это было до того, как я изменил его на пирог) на красном фоне, окруженный желтой рамкой, увенчанный красным, заголовок- с анимацией все работает так, как надо
- Когда я изменил frameflag = 3 для frameflag = 2 точка вращается вокруг в нижнем левом углу, и, когда я изменил снова Scilab сказал, что ручка не действует больше. Просто зайдите и перезагрузите
- Я также видел размер изменения графика Window от исполнения к исполнению без всякой видимой причины Короче говоря, эти события дают ощущение, что анимация- вместе с ГПИ-не является главным приоритетом команды Scilab

Demo 2 (1/2): moving rectangles

- Это демо составлена на основе Chancelier др..
- Это попытка продемонстрировать использование команды XOR в f.pixel_drawing_mode = 'xor', но, не вместо NOR по причинам, указанным ниже.
- Прямоугольники перемещаются в верхней части на сером фоне
  Прямоугольники нарисованы с xfrect () без цветового кода, поэтому они черные
  Прямоугольники двигаются по диагонали из угла в угол в 200 шагов

```
// animation_rectangles.sce

// Two rectangles slide diagonally over the Graphics /
// Window. As they slide over each other their colors /
// are NORed. The solution is only partly successful /

clear,clc,clf();

f=gcf();
f.pixmap='on';        // Double buffer mode
f.pixel_drawing_mode='nor';        // NOR mode
f.background=color("lightblue");

ax=gca();
ax.data_bounds=[0,-4;14,10];        // Plot limits
ax.margins=[.1 .1 .1 .1];              // Plot framed
ax.background=color("lightgrey");
max_pos = 10;       // Max position of rectangles

k=%nan;              // Auxiliary parameter
xfrect(k,k,4,4);       // First black rectangle
e1 = gce();
xfrect(max_pos-k,max_pos-k,4,4);  // Second rectangle
e2=gce();

for k=linspace(1,10,200)              // Animation loop
   e1.data(1:2)=k;
   e2.data(1:2)=max_pos-k;
   show_pixmap()                 //Show double buffer
end
```

Demo 2 (2/2):замороженные участки



Вот анимация в стадии разработки. НИ функция делает свою работу, но в остальном что-то совершенно неправильное:
Мы не имеем черные прямоугольники, перемещаемых на светло-сером фоне

Проблема в том, что команда f.pixel_drawing_mode = ', ни' работает на весь экран, а не только на движущиеся прямоугольники, согласно предписанию Chancelier др.. По этой причине операция XOR они используют еще хуже, чем NOR

Я решил оставить демо в этом состоянии. Те, кто заинтересован, могут найти лучшее решение в Стира Scilab Graphics, с. 28

Демо 3 (1/3): 3D-объект, скрипт (1/2)

- Теперь мы будем смотреть на более сложные геометрические объекты, 3D-графики, которые движется и по азимуту и вокруг оси
- Границы данных не определяется отдельно, они меняются с разрешением поверхности сетки.
- Первая команда графика только определяет название осей

```
// rotating_surface.sce

// The 3D surface is first rotated and then   /
// tilted, after which its position is locked    /

clear,clc,clf;

// Initialize:
//-------------
f=gcf();
f.pixmap="on";
clear_pixmap();
t=%pi/20*(-20:20);   // Bounds & mesh resolution

// First plot command, defines labels:
//-------------------------------------------
plot3d1(t,t,sin(t)'*cos(t),%nan,%nan,..
    'x_axis@y_axis@z_axis');
```

Демо 3 (2/3): 3D-объект, скрипт (2/2)

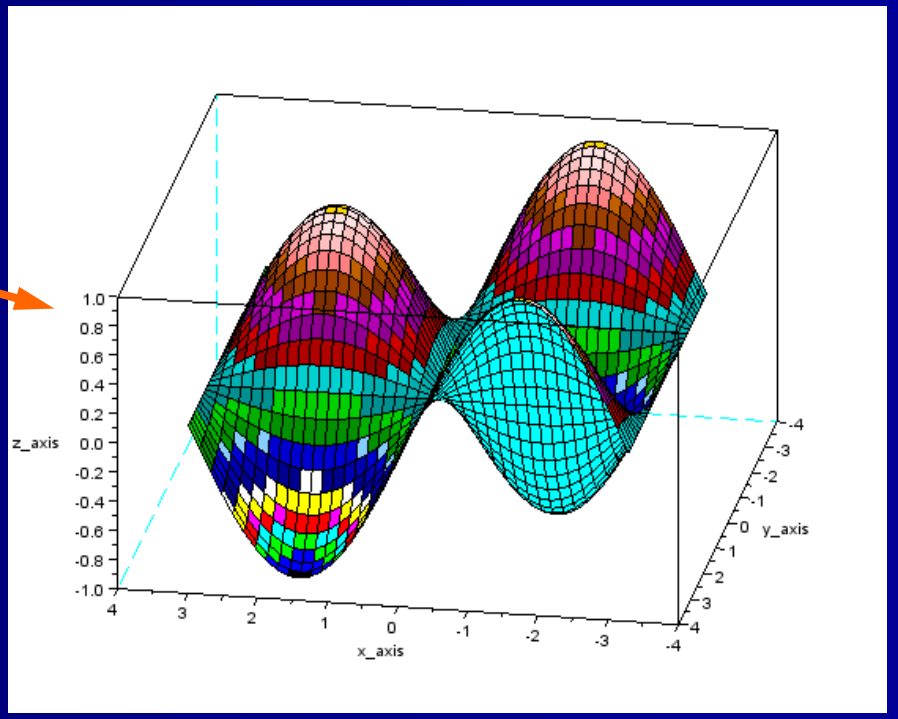
- Поверхность вращается вокруг оси z, начиная с 45 º и заканчивая 100 º, а угол наклона постоянный на 45 º
- Когда перестает вращаться, поверхность наклоняется вокруг оси x от 45 º до 80 º, с углом постоянным 100 º
- С моим ноутбуком 1,6 ГГц анимация отлично не работает, заметны скачки от шага к шагу

```
// Set speed and turn object:
//-------------------------------------
step = 2;              // Step size --> 1/speed
for angle1 = 25:step:100,     // Rotate loop
    plot3d1(t,t,sin(t)'*cos(t),angle1,45)
    show_pixmap();
end
for angle2 = 45:step:80,      // Tilt loop
    plot3d1(t,t,sin(t)'*cos(t),100,angle2)
    show_pixmap();
end
f.pixmap="off";
```

Демо 3 (3/3): 3D-объект, участок

Поверхность достигла своего назначения: поворачивается на 100 ° (азимут) и наклонять до 80 º (высота)

Во время тестирования различных параметров я увидел это сообщение на консоли (длинный список). Он исчез, когда я повторно запустить сценарий




```
Exception in thread "AWT-EventQueue-0" java.lang.NullPointerException

        at javax.swing.plaf.basic.BasicTextUI$RootView.paint(Unknown Source)
        .....
```

Dr. EW
Johnny Heikell

18. Разное

Каша философии и реализма, которую, мы надеемся использовать

Процесс решения проблем

Процесс решения проблем в вычислительной задаче обычно проходит через следующие этапы:

1. Определение проблемы (ответить на вопрос "В чем проблема?")
2. Определение путей решения проблемы (блок-схема, DFD и т.д.)
3. Определение уравнения и / или алгоритмов (решить математическую задачу)
4. Трансформировать шаги 2 и 3 к программной структуре
5. В кодирование с шагом проверить каждый шаг, прежде чем приступить
6. Подтвердить решение

Границы между этими шагами могут быть размыты, итераций в основном необходимы, и один или два из шагов может быть более важным, чем другие. Каждый шаг требует также ряд подзадач, которые должны выполняться. Но в целом этот подход помогает при решение проблем.

Хорошие структуры программы

- Имейте в виду, что программа характеризуется его структурой и ее возможностью
- Давайте переменным четкие и осмысленные имена; использовать отдельные буквы только для X, Y и Z., счетчиков циклов (J, K) и т.п.
- Сплит код в логических объектах с помощью подпрограмм
- Обособленные структурных лиц по пустым строкам и заголовкам комментариев
- Отступ строк (петли, команды печати и т.д.) для повышения ясности
- Будьте либеральны с использованием комментариев, держать пространство между командой и ее комментарием
- Простой красиво; хорошая программа короткая и недвусмысленная
- Для более тщательного обсуждения см. учебники по программной инженерии

Подводные камни: Программирование (1/4)

- Компьютерные программы не могут гарантировать 100% надежность. Существует опасность как скрытых, так и очевидных ошибок.
- Основные учебники Scilab не обращают внимание на программные ошибки.
- «Eine Einführung в Scilab» Бруно Пинко (оригинал на французском языке, является исключением. В его последней главе кратко рассматриваются ошибки программирования.
- Стоит поискать в Интернете «Matlab подводные камни», где предоставляются некоторые намеки.
- Существует также полезная дискуссия в главе 9 Хан, Валентина:. «Основные Matlab для инженеров и ученых», 3-е изд, Баттерворта Гейне Манн, 2007

Подводные камни Программирование (2/4): типы ошибок

Ошибки программирования в целом можно разделить на следующие виды:

- Логические (ошибки в алгоритме)

  Логические ошибки являются в основном результатом нашей ограниченностью понимания проблемы или нашим ограниченным знанием алгоритмов в целом и Scilab, в частности

- Синтаксические (ошибки в сценарии Scilab)

  Синтаксические ошибки появляются из-за человеческих ограничений: надзор, небрежность, забывчивость, и тому подобное. Типичные случаи неправильное написание, несоответствие кавычек, неправильный тип аргументов и т.д.

- Округление (ошибки, связанные с ограниченной точностью компьютера)

  Ошибки округления возникают из усечений а аппаратных ограничений, оцифровке математических функций, преобразования между десятичным и двоичным кодами и т.д.

- Существует также четвертый тип, а именно ошибки, допущенные в систему дизайнерами и программистами Scilab. Они показывают, ограничения производительности, бедные пользовательские интерфейсы.

Подводные камни Программирование (3/4): сообщения об ошибках

"Несовместимые длины векторов "был бы лучшим сообщением об ошибке

```
-->[1 2 3] + [4 5]
          !--error 8

Inconsistent addition.
```

```
-->[1 2 3] * [4 5 6]
          !--error 10

Inconsistent multiplication.
```

Это сообщение приводит в заблуждение, если то, что вы намерены [] '* [], но хорошо, если вы нацелены на поэлементного умножения []. * [] (Но "Неправильный умножение« все же лучше)

```
-->sqrt = 5^2 + 3*17
Warning : redefining function: sqrt                    . Use funcprot(0) to avoid this message

 sqrt  =

    76.
```

Здесь вы можете увидеть, что предупреждение "переопределение функции" действительно имеет смысл. Я неправильно использовал SQRT(как имя переменной), но Scilab признает это как встроенная функция. Ответ правильный, но надо скорее изменить имя переменной. Проверьте имя, если вы не уверены, если предназначения данного имя переменной защищены.

Подводные камни программирования (4/4): бесконечный цикл

Я уже несколько раз упоминал риск создания бесконечного цикла, так что давайте посмотрим на «этого маленького зверя»

При выполнении сценария вы должны остановить исполнение программы, чтобы остановить его. Самый простой способ, это нажать на кнопку «закрыть» на консоли, а затем перезагрузить Scilab

Почему цикл не заканчивается? Потому что мы умрем от старости, прежде чем переменная n случайно получит значение, равное 0,5

```
// endless_loop.sce

// Demonstrates an endless loop.    /
// Execution ends only by crashing  /
// the program (click on the Close  /
// button (X) on the Console)       /

n = .1;
dt = getdate();
rand('seed',1000*dt(9) + dt(10));
while n ~=0.5;
    n = rand(0,'normal');
end;
disp(n)
```

Вы забыли о записи функции RAND? Если это так, вернитесь к Примеру 1-3 (лото ничья)

Отладка (1/2)

- Мы уже знакомы с элементами встроенного отладчика Scilab, который предоставляет сообщения об ошибках на консоли (отдельное окно отладчика может прийти с Scilab 6.0)
- Еще один инструмент отладки является пауза, возобновление, прервать набор операторов. Читайте в разделе 6.7 в Введение в Scilab Майкл Боден для объяснения
- Мое предложение для безболезненного программирования поэтапная разработка, выражается в следующем:

  Разработать сценарий наизнанку, начиная с центрального уравнения (или подобного «ядра») к выполнению его с помощью простого сюжет и команд отображения. Исправьте "ядро", пока оно работает удовлетворительно.

  Продолжайте сценарий скрипт поэтапно, добавляя подпрограммы, петли, команды сюжет, обрабатывая команды, и т.д. и проверяйте (выполнения) после каждого добавленного шага
- Преимущество поэтапного развития заключается в том, что, во-первых, ошибки изолированные в определенной части сценария легко определить и, во-вторых, каждый получает чувство удовлетворения от каждого шага, выполняющегося без ошибки.

Отладка (2/2): проверка

- Наконец, даже когда сценарий кажется, правильным мы должны проверить его. «Не судите птицу по цвету его перьев»

Для проверки вы можете:

- Критически взглянуть на решения: Это логично звучи, вы знаете, что программа делает, а что - нет?
- Проверьте и устраните неполадки(я нашел их, что удивительно, во многих примерах учебников, которые я использовал)
- Запустите программу для некоторых случаев, на которые вы знаете ответ. Если такие случаи отсутствуют, проверьте по крайней мере, что ответы, которые выдает программа, правдоподобны и величины верны.
- Тест на "необычных события" (например, где вы могли бы в конечном итоге делить на ноль), экстремальные значения (например, бесконечность), условия, приводящие к петле блокировки, и т.д.
- Работа в рамках программы, вручную, чтобы увидеть, если вы можете определить, где вещи могли бы начать происходить не так
- Попросите кого-нибудь умнее, чем вы, на второй взгляд.

Ускорение Scilab (1/4): введение

- Есть способы, чтобы ускорить выполнение программ Scilab. Три основных правила:
  - Заменить петли на векторные. * Особенно с цикл `for` стремиться к его векторизованного альтернативы
  - Используйте подпрограммы по возможности
  - Избегайте трудоемкие алгоритмы, как Рунге-Куттf
- Ускорение, особенно если мы будем двигаться с петель, чтобы векторных функций требует, чтобы мы приняли новое мышление. Это требует усилия в обучения. Но векторы, в конце концов, это все о Scilab!
- Тем не менее, существует проблема с изучением векторных операции: Учебники говорят нам, использовать их, но удаляют мало внимания на данную тему, и дают всего несколько примеров, и то, только основных.

*) Scilab браузер не поддерживает функцию Matlab `vectorize()`.

Ускорение Scilab (2/4): векторные основы функции

- Этот случай составлена на основе Бодена. Задача состоит в том, чтобы вычислить сумму нечетных чисел [1,99]?
- В первом случае мы используем while...if...end...end сценарий, выбирая нечетные числа modulo() функцией.
- Ниже приводится альтернативное векторное решение. Чистый и простой! Преимущества:
- На языке высокого уровня, легче понять
- Выполняет быстрее, чем с большими матрицами

```
// add_demo1.sce

clc;
add = 0;
i = 0;
while ( i < 100 )
    i = i + 1;
    if ( modulo( i, 2 ) == 0 )  then
        continue;
    end
    add = add + i;
end
disp( add )
```

```
2500.
```

```
// add_demo2.sce

clc;
add = sum(1:2:100);
disp(add)
```

```
2500.
```

Существовала ошибка в Scilab 5.3.1 и он вернулся сообщение об ошибке «Неправильный индекс» для последнего скрипта

Ускорение Scilab (3/4): время выполнения tic()..toc()

- Время генерации может быть измерена парой функций tic()..toc()
- Верхний сценарий вычисляет. Значения за sin(x) и показывает результат в таблице из двух столбцов (показан в крайнем правом приложение только для четырех точек). Время выполнения 17,389 сек для показанного сценария, с Scilab цикла по 30000 раз
- Нижний (Векторная) сценарий выполняет ту же задачу. Время выполнения составляет 9 мс, примерно в 2000 раз быстрее, чем с циклом `for...end`!

```
// measure_time1.sce

clear,clc;
x=[];      // Initate vector
y=[];      // Ditto
tic();     // Start stopwatch
for t=0:0.0002:2*%pi
    x=[x; t];
    y=[y; sin(t)];
end
time=toc();  // Stop watch
disp(time)   // Display time
```

```
0.    0.
2.    0.9092974
4.  - 0.7568025
6.  - 0.2794155
0.014
```

```
17.389
```

```
// measure_time2.sce

clear,clc;
tic();
t = (0:0.0002:2*%pi)';
[t,sin(t)]
disp(toc())
```

```
0.009
```

Ускорение Scilab (4/4): еще две идеи

Замена цикла на `ones()`

```
tic();
for i = 1:100000
    x(i) = 1;
end
disp(toc())
```

77.481

```
tic();
x = ones(100000,1);
disp(toc())
```

0.005

Замена вложенных циклов с длиной `length(find())`:

```
tic();
k = 0;
for i = 1:1000000
   x = rand(1,1);
      if x < 0.2 then
         k = k + 1;
      end
end
disp(['k =' string(k)])
disp(['time =' string(toc())])
```

!k = 200660 !

!time = 10.142 !

```
tic();
k = length(find(rand(1000000,1) < 0.2));
disp(['k =' string(k)])
disp(['time =' string(toc())])
```

!k = 199649 !

!time = 0.298 !

В этом случае время выполнения уменьшается на коэффициент 34. Не так много, как в более ранних случаях, но все же значительное улучшение (типичное в практике)

Несоответствие в измерениях времени (1/2)

Я хотел проверить время вычислений Scilab для как это представлено в учебнике по Matlab Ганом и Валентина. Сначала я сделал это в консоли, а затем в редакторе, но результаты не совпадают:

```
-->tic();
-->s = 0;
-->for n = 1:100000
-->s = s + n;
-->end
-->time = toc();

-->disp(time)


   97.531
```

В результате 97,531 секунд на консоли. Ясно, что не верно, потому что ответ пришел незамедлительно

0.453 сек когда делал в редакторе. Это больше походит на правду

```
// scilab-matlab_loop.sce

clc;
tic();
s = 0;
for n = 1:100000
   s = s + n;
end
time = toc();
disp(time)
```

```
   0.453
```

Давайте попробуем с векторной функцией (следующий слайд)

Несоответствие в измерения времени (2/2)

И то же самое в векторной форме :

```
-->tic();
-->n = 1:100000;
-->s = sum(n);

-->time = toc();

-->disp(time)

   32.994
```

Теперь консоль рассказывает о улучшения времени вычислений в 3 раза, но все же это не так ..

и редактор соглашается об улучшении, но расхождение остается

```
// scilab-matlab_vectorized.sce

clc;
tic();
n = 1:100000;
s = sum(n);
time = toc();
disp(time)
```

```
0.016
```

Заключение: Существует ошибка либо в моем подходе или в Scilab; но Scilab кажется выполняет быстрее, чем Matlab на старом процессоре Pentium II, который использовали Хан и Валентина

ATOMS (1/6): установка новых наборов инструментов

- Напомним, проблемы с атомами, я упоминал в главе 1
- Атомы (Модуль автоматического управления для Scilab) позволяет пользователю загрузить и установить внешние наборы инструментов (модули)
- Существует причина, чтобы узнать, какие модули могут быть полезны, так как специализированные инструменты могут ограничить время, необходимое для решения проблемы
- Начните с подключения компьютера к Интернету и нажмите на иконку атомов на консоли. Если вам не повезло, вы увидите следующее сообщение на консоли

```
atomsDownload: The following file hasn't been downloaded:
    - URL            : 'http://atoms.scilab.org/5.3/TOOLBOXES/32/windows.gz‘
    - Local location : 'C:\Users\Johnny\AppData\Local\Temp\SCI_TMP_2772_\.atoms\1_TOOLBOXES.gz‘
```

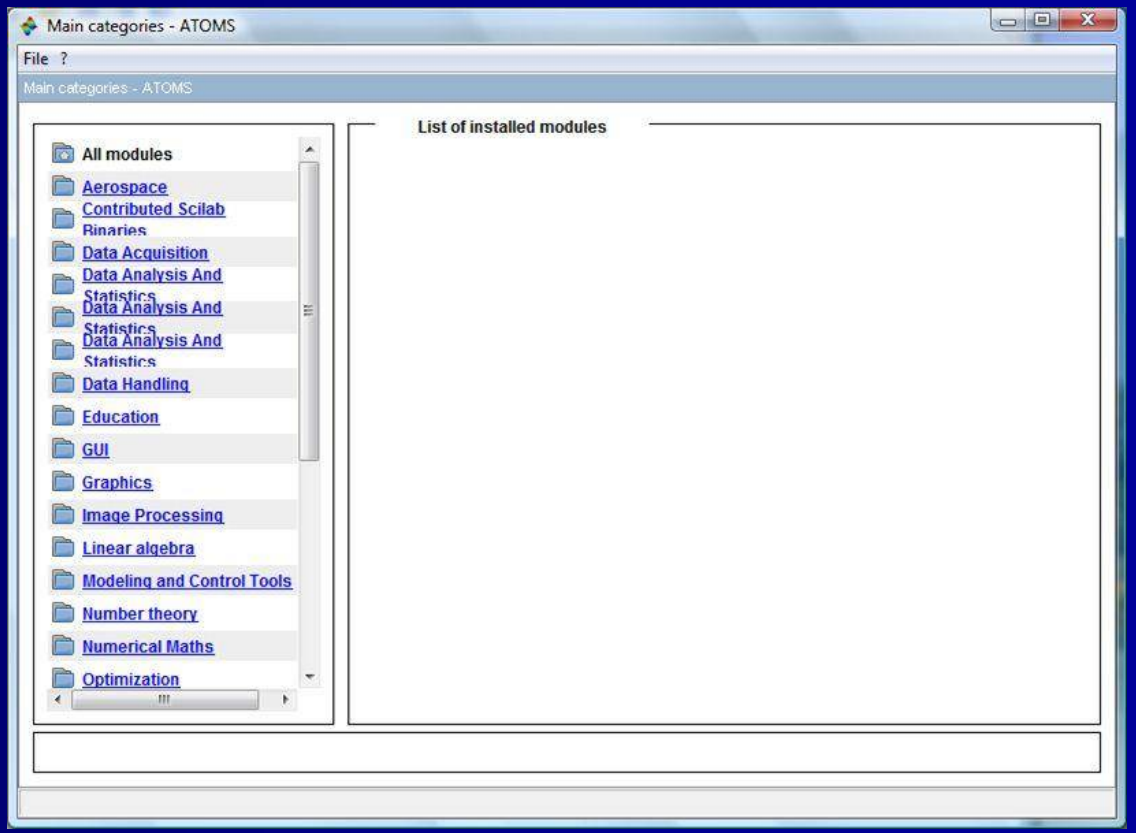
- Ошибка Scilab (ошибка # 8942) остается нерешенной и его истинное значение мне неизвестно. Команда Scilab дает бесполезный предложение загрузить указанный файл

ATOMS(2/6): то, что доступно

Это главное окно **ATOMS**. Тексты немного пересекаются, но в основном это список содержимого.

Идем дальше и попытаемся найти что-то интересное, даже если из нас не так много инженеров

Еще одна проблема в том, что существует очень мало информации о том, что действительно могут сделать для нас данные модули.

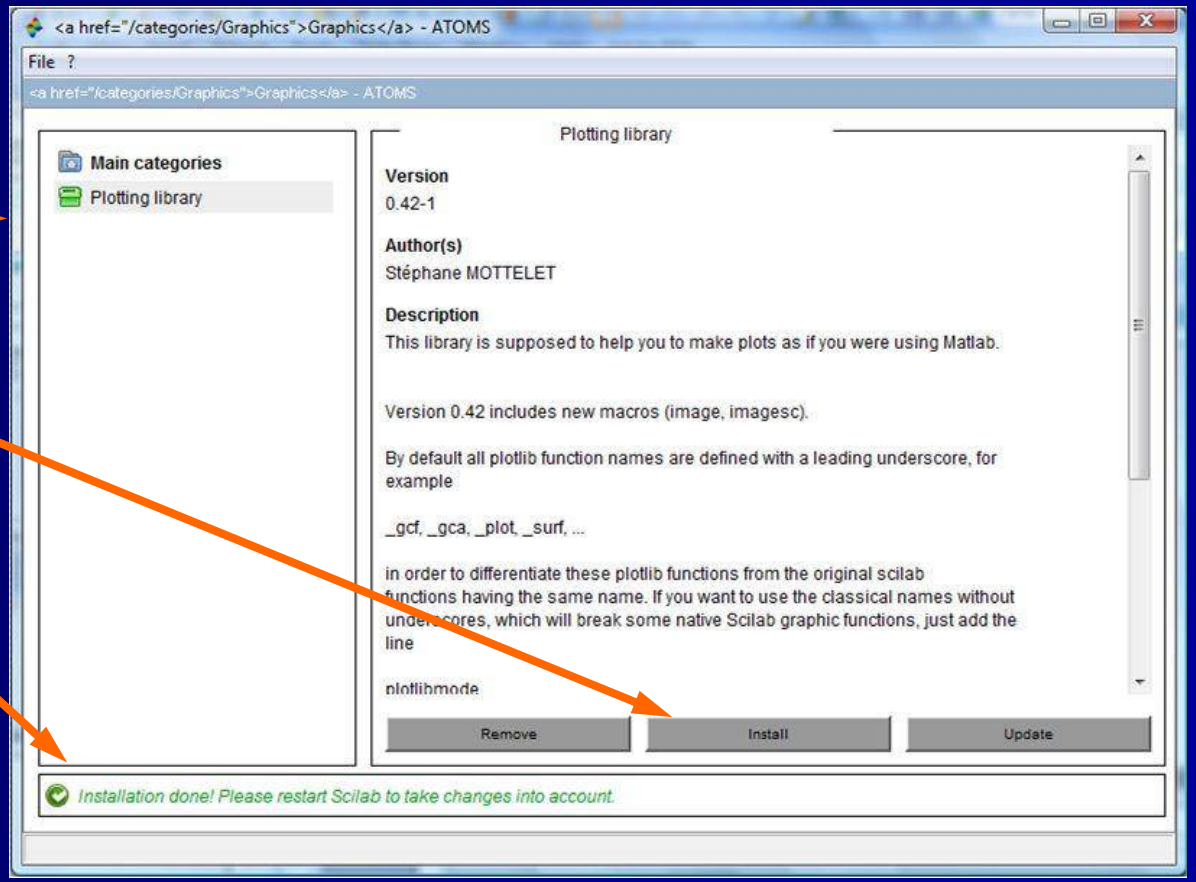


ATOMS (3/6):установки

Я решил попробовать "построение библиотеки" Стефани Mottelett (это версия имеет проблемы с Vista PC!)

Нажмите на Установить

Сообщение установка открывается в нижней части, и после довольно долгое время говорит Scilab, какаой был установлен модуль



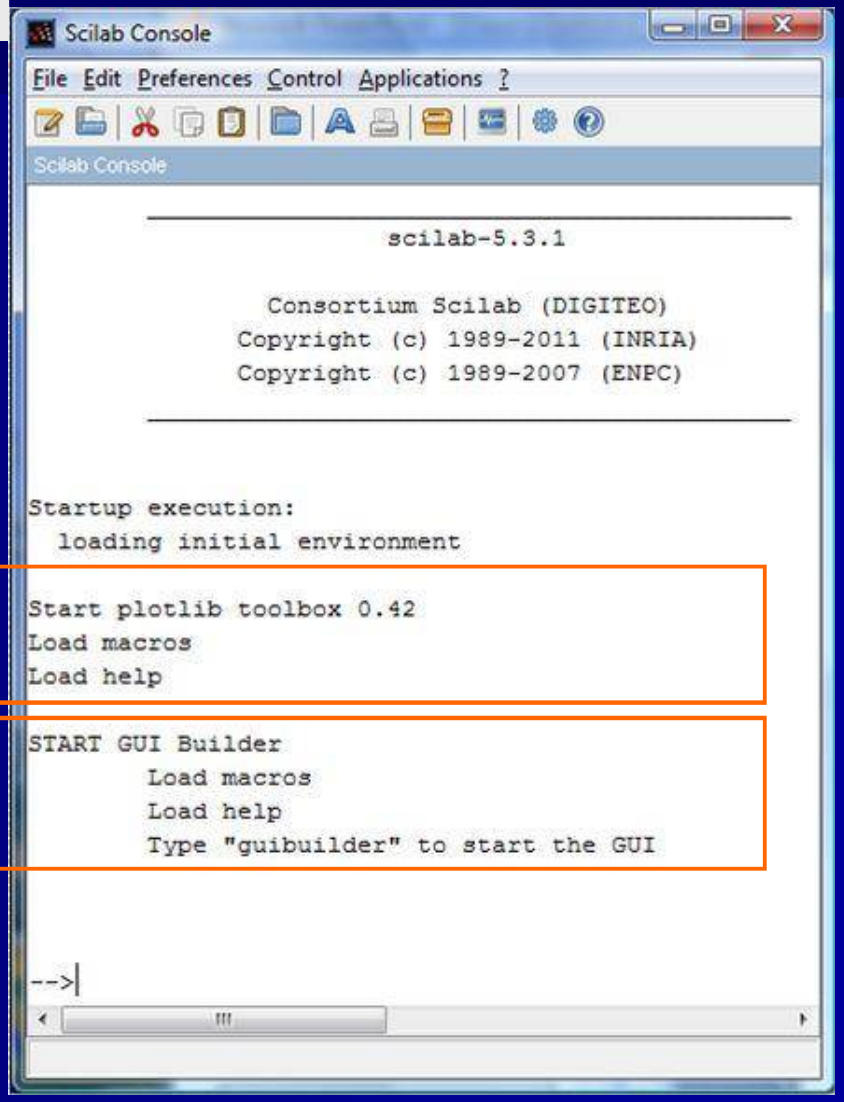
ATOMS (4/6): новая информация

Я также установил GUI Builder по Тан Чин Лух

Когда Scilab перезапустил, об этом сообщает установленная панель инструментов

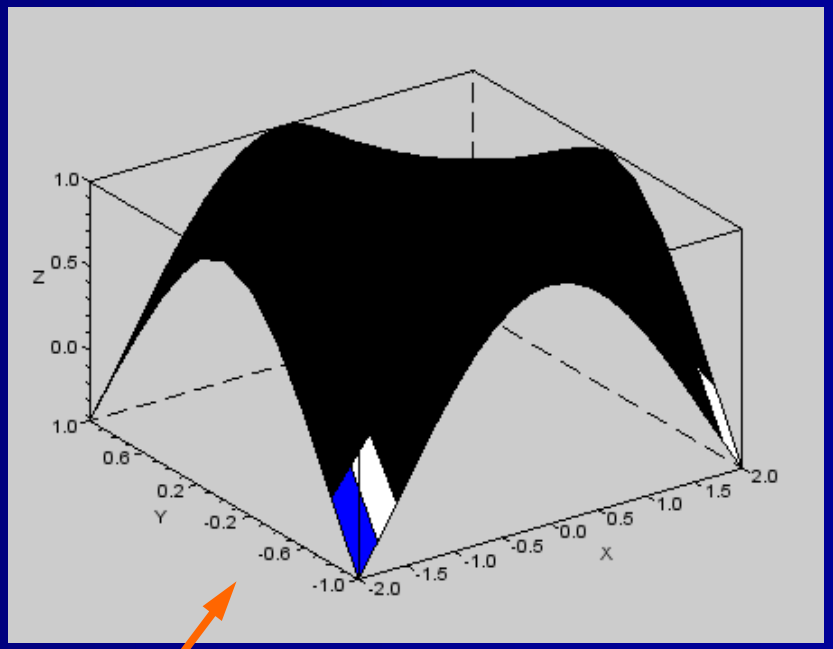
Вопрос: Что нужно, чтобы воспользоваться установленными модулями?

Проверьте с помощью браузера, в самом конце списка содержимого новые дополнения: "Matlab, как построение библиотеки» и «Графический интерфейс пользователя Builder”



ATOMS (5/6): Проверьте с Matlab'S quiver3()

```
// matlab_quiver3.sce

// Test Matlab's quiver3() function /

clear,clc,clf;

[X,Y]=meshgrid(-2:0.2:2,-1:0.2:1);
Z=cos(X.*Y);
surf(X,Y,Z);
hold on
   [U,V,W] = surfnorm(X,Y,Z);
   quiver3(X,Y,Z,U,V,W,'r');
   Legend 'Surface normals'
   colormap gray
hold off
```



Я тестировал функцию Matlab в quiver3 () со сценарием в справке, но что-то не так. Сюжет не такой, и Scilab «кричит».

```
 hold on
 !--error 4
Undefined variable: hold
at line      10 of exec file called by :
es\matlab_quiver3.sce', -1
```

ATOMS (6/6):обсуждение проблем

- Мои неприятности начались всерьез с исполнением quiver3 от Matlab функции (). Независимо от того, что я сделал, все скрипты Scilab не оказалось мусора участки
- Был смущен некоторыми событиями: Помимо наборов инструментов я также установил Scilab 5.3.2, что некоторые обновления для Windows, прибывающим, и увидел подвисание как с MS Word и Windows. Такой проблемы не было с Scilab 5.1.1
- Окна были запущены в течение трех лет, поэтому я решил переустановить его. Только после этого процесса я знал про ATOMS
- Проблемы, связанные с инструментом Plotlib. Я удалил его и Scilab 5.3.2 снова нормально заработал
- Извлеченные уроки: Устанавливайте только один набор инструментов, и сразу проверяйте его и Scilab сразу. Удалите инструмент в случае возникновения проблем

Построение библиотеки сценария

- Со временем мы накапливаем огромное количество программ . Как мы должны управлять ими , как мы можем позже найти то, что нам нужно?
- Эта презентация демонстрирует альтернативные способы комментируя скриптами - самый важный вопрос , когда программа должна быть изменена в будущем
- Обратите внимание на названия программ . Описательные названия помогут выявить индивидуальные программы среди других программ в большом списке файлов.
- Создайте свою библиотеку сценариев Scilab в логическом порядке . В этой работе я частично сохранены скрипты на джойстик привода , в файле H: \ Dr.EW \ Писаний \ примеры Scilab \ , в предположении, что эта презентация указывает на , где найти тот или иной сценарий.
- Один из вариантов заключается в поддержании таблицы каталога программ с информацией о том, что конкретный скрипт , в котором он находится , который функционирует в нем содержится и т.д. передовое решение документации программное обеспечение типа используемого в управление требованиями

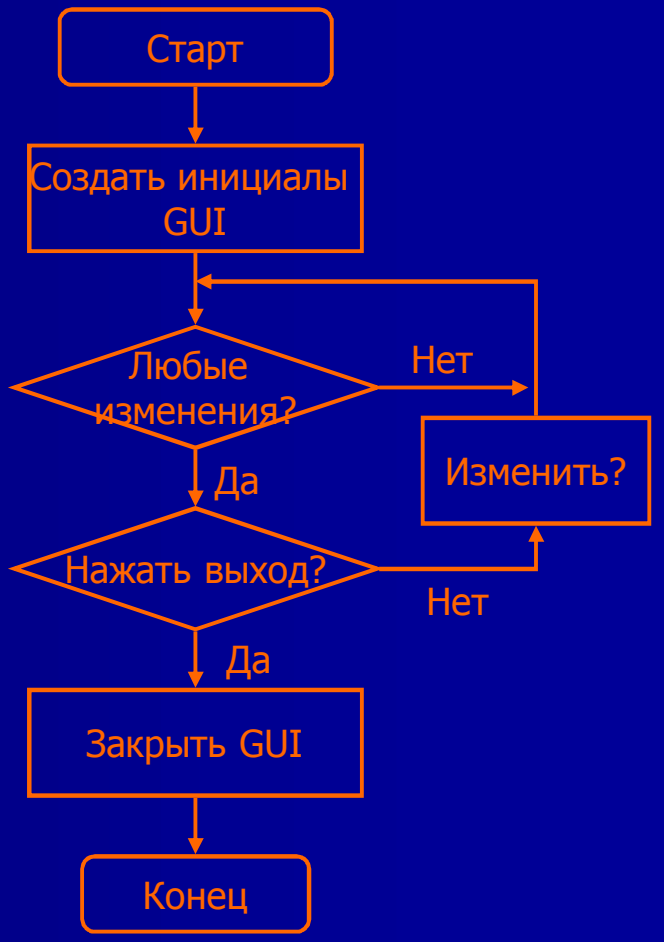
Dr. EW
Johnny Heikell

19. Примеры №6

Дополнительные примеры, в основном, связанные с главами 15-19

Пример 6-1: определение пользователем GUI, введение

- Этот пример является модификацией похожи A. Chiaverini: Введение Scilab 5.3, стр. 74-80
- Задача состоит в том, чтобы создать графический интерфейс в окне с графикой (GW). Графический интерфейс состоит из
  - Синусойда
  - Слайдер для изменения угловой частоты сценария
  - Две «RadioButton» при которых свойства проложенного графика могут быть изменены
  - Кнопка выхода, что закрывает интерфейс
- Процесс показан справа в виде блок-схемы



Пример 6-1: определение пользователя GUI, сценарий

Первая подпрограмма, initial_GUI (), создает начальное значение графика для синуса; включая название и осей этикетки

Начальная угловая частота ω определяется как 5 Гц

Здесь, на самом деле, ничего особенного.

```
// GUI_italian.sce

// Generates on the Graphics Window (GW) a GUI that contains a  /
// sine plot (plot2d), a slider by which to adjust the angular  /
// frequency of the sine function, two radiobuttons that change /
// the style and color of the sine graph, and a pushbutton that /
// closes the GW                                                /

clear,clc;

// **** SUBROUTINES **** //

// Declaration of initial plot in GUI:
//---------------------------- ---------------
function initial_GUI()
   t = linspace(0,7,200);
   w = 5;                                  // Initial angular frequency
   plot2d(t,sin(w.*t),..                   // Initial plot w=5 rad/s
      rect = [0,-1.1,7,1.1]);
   a = gca();
   a.axes_bounds = [0.2,0,.8,1];     // Frame dimensions & location
   xtitle("GUI DEMO WITH sin (wt)",...
      "Time [s]","Amplitude");
   a.font_size = 3;                        // Axes mark size
   a.x_label.font_size = 3;                // x_label size
   a.y_label.font_size = 3;                // y_label size
   a.title.font_size = 3;                  // Title size
endfunction
```

Пример 6-1: ... сценарий/...

Следующие две подпрограммы реагируют на команды пользователя (слайдер и Radiobuttons соответственно), и указывают на Fourt подпрограммы, new_GUI_data ()

Существующий участок стирается

Слайдер идет от одного конца до другого в 10 шагов

В if-then-else-end конструкции зафиксируйте статус в зависимости от того какая кнопка была нажата

```
// Functions for changes wrt user actions:
//-------------------------------------------------
function update_slider()          // IF slider movement
   new_GUI_data();                // GOTO new_GUI_data()
endfunction

function update_radio()           // IF radiobutton click
   new_GUI_data();                // GOTO new_GUI_data()
endfunction

// Redefine plot in GUI:
//-----------------------------
function new_GUI_data()
t = linspace(0,7,200)
drawlater();                      // Delay changes
   a = gca();
   if (a.children~=[]) then       // IF frame contains graph...
      delete(a.children);         //    then delete graph
   end
   w = h_slider.value/10;         // Slider range: 10 steps
   plot2d(t,sin(w.*t));
   if (h_radio1.value == 0) then  // Check status of style button
      a.children.children.polyline_style=1;   // Basic style: line
    else
      a.children.children.polyline_style=3;   // IF clicked: bars
   end
   if h_radio2.value==0 then      // Check status of color button
      a.children.children.foreground=1;       // Basic color: black
    else
      a.children.children.foreground=2;       // IF clicked: blue
   end
drawnow();
endfunction
```

Пример 6-1: определение пользователя GUI, сценарий

Основная программа сначала удаляет существующий ГВт

Размер и расположение нового GW определяется как функция от общего размера экрана

Здесь мы возьмем первоначальный сюжет, который поставляется в ГВт (GUI)

Следующее, что мы добавляем к GUI кнопка EXIT.

```
// **** MAIN **** //

xdel();
funcprot(0);

// Define window size & position:
//---------------------------------------
screen_size = get(0,"screensize_px");  // Find computer screen size
size_x = .7*screen_size(3);            // .7*screensize_px 3rd element
size_y = .7*screen_size(4);            // .7*screensize_px 4th element
h_graph = scf(0);                      // Open Graphics Window
h_graph.figure_size = [size_x size_y]; // Define GW size
h_graph.figure_position =...           // Position GW in the...
   [size_x/5 size_y/6];                // middle of the screen

// Open GUI with inital plot:
//-----------------------------------
initial_GUI();

// Add EXIT button:
//-----------------------
h_stop = uicontrol (h_graph,...
   "style","pushbutton",...            // Declare pushbutton
   "string","EXIT",...                 // Pushbutton label
   "fontsize",14,...
   "backgroundColor",[1 0 0],...       // Red button RGB
   "foregroundColor",[1 1 1],...       // White label RGB
   "position",[85 size_y-210 50 50],...
   "callback","xdel(0)");              // CLOSE GW if button pushed
```

Пример 6-1: определение пользователя GUI, сценарий

Вот команда UIControl (), которая управляет ползунком

Strcat () является функцией, мы не встречались раньше. Обратите внимание, что w и rad/s окружены двумя ковычками (" W " и " рад / с "), а не апострофами

Это начальный вид под ползунком

И функция UIControl (), которая заботится об изменениях меток

```
// Add slider & label:
//-----------------------
h_slider= uicontrol(h_graph,...
   "style","slider",...                          // Declare slider
   "Min",0,...                                   // Slider start value
   "Max",100,...                                 // Slider end value
   "value",50,...                                // Initial slider value
   "position",[10 size_y-270 180 20],...         // Slider size & location
   "callback","update_slider();...  // GOTO to update_slider()
   foo = strcat([ ' ' w = ' ' string(h_slider.value/10)...
      ' ' rad/s ' ']);h_text_slider.string = foo");
slidelbl = strcat(["w = 5 rad/s"]);              // Define initial label
h_text_slider = uicontrol(h_graph,...
   "style","text",...                            // Declare text
   "horizontalalignment","center",...            // Position in reserved field
   "string",slidelbl,...                         // Add slider label
   "fontsize",14,...
   "backgroundColor",[1 1 1],...                 // White background
   "position",[10 size_y-310 180 20]);           // Field size & location
```

Пример 6-1: определение пользователя GUI, сценарий (5/6)

Первая RadioButton контролирует стиль проложенной синусоиды (сплошная линия используется по умолчанию, превращается в гистограмму, когда RadioButton нажата)

Команды очень похожи на те, что в слайдере, единственное, что команда Foo отсутствует

```
// Add radiobutton for graph style:
//-----------------------------------------
h_radio1 = uicontrol(h_graph,...
   "style","radiobutton",...                  // Declare radiobutton
   "Min",0,...
   "Max",1,...
   "value",0,...                              // Initial button value
   "backgroundColor",[1 1 1],...
   "position",[10 size_y-350 20 20],...
   "callback","update_radio()");              // GOTO to update_radio()
h_text_radio1 = uicontrol(h_graph,...
   "style","text",...                         // Declare button text
   "horizontalalignment","left",...
   "string","-Change graph style",...
   "backgroundColor",[.8 .8 .8],...           // Gray background
   "fontsize",14,...
   "position",[40 size_y-350 140 25]);        // Field size & location
```

Пример 6-1: определение пользователя GUI, сценарий(6/6)

Второй RadioButton управляет цветом построенной синусоиды (черный цвет по умолчанию, превращается в синий, когда RadioButton нажата)

Это в основном повторением команд для первого RadioButton, но позиция в GW отличается

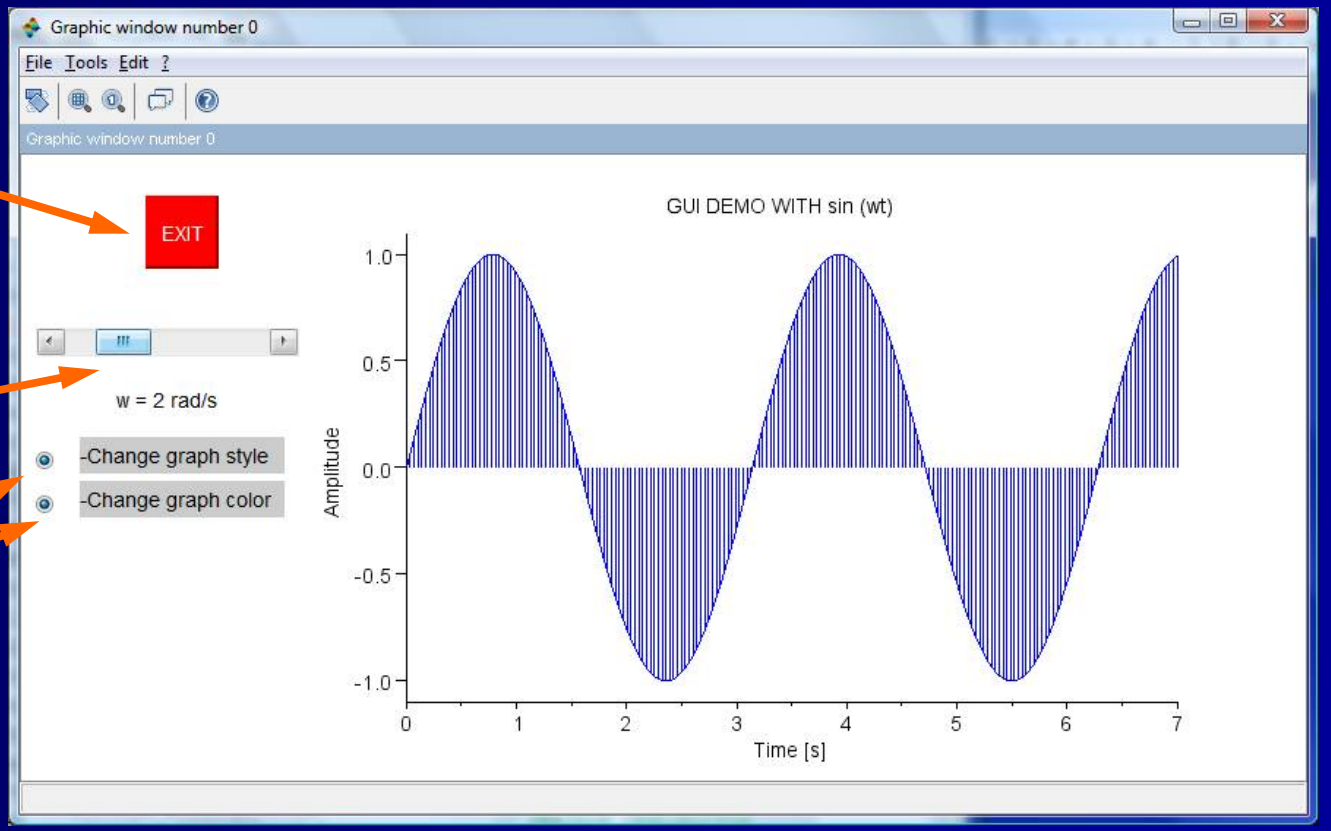
```
// Add radiobutton for graph color:
//-----------------------------------------
h_radio2 = uicontrol(h_graph,...
   "style","radiobutton",...                  // Declare radiobutton
   "Min",0,...
   "Max",1,...
   "value",0,...                              // Initial button value
   "backgroundColor",[1 1 1],...
   "position",[10 size_y-380 20 20],...
   "callback","update_radio()");              // GOTO to update_radio()
h_radio2_txt = uicontrol(h_graph,...
   "style","text",...                         // Declare button text
   "horizontalalignment","left",...
   "string","-Change graph color",...
   "backgroundColor",[.8 .8 .8],...           // Gray background
   "fontsize",14,...
   "position",[40 size_y-380 140 25]);        // Field size & location

// **** END MAIN **** //
```

Пример 6-1: Определение пользователем GUI

Нажмите на EXIT и окно закрывается

W нажал до 2 рад / с

Обе радиокнопки были нажали



Проблема: Scilab испытывает замок, если вы перетащите бегунок. Сюжет замерзает и Консоль не сообщает, что текущий дескриптор больше не существует

Ex 6-1: обсуждение

- Я копирую - вставляю скрипт из Антонелли и Chiaverini в редактор Scilab
- Сценарии должны были быть очищены и некоторая избыточность должна быть удалена
- Я добавил второй RadioButton и организовал сценарий в том, что я думал, было более логично.
- Когда я выполнил сценарий он открылся, как и ожидалось, но слайдер не хватает
- После разочарований я сделал все с самого начала, но теперь пошагово. Ошибка в том, что я перешел из `if-then-end в` конструкцию функции new_GUI_data () после команды plot2d()
- Уроки извлечены: Сделайте ступенчатую работу и испытайте, вы заметите прогресс
- Что касается блокировки, я думаю, что Scilab запускается в конфликтной ситуации, когда он должен обновить ручку и в процессе предыдущего обновления

Пример 6-2: анимация из Waltzing многоугольника (1/4)

- Это демо основана на Pincon в "Eine Einführung в Scilab»
- Оригинал содержал ошибки, устаревшие функции и избыточные команды. Например, я преобразовал Xset () функцию для обработки графических команд (как описано в главе 7)

```
// animation_pincon_m2.sce

//-------------------------------------------------------------------------/
// The script plots the track of a blue polygon (rectangle)               /
// with red border, as it turns around its axis while racing              /
// counterclockwise in a circular loop on a black background.             /
// The rectangle can be chaged to a trapetzoid or other shape             /
// by changing element values in the matrix polygon. Changing             /
// theta arguments in the matrix align gives different effects            /
//-------------------------------------------------------------------------/

clear,clc,clf;

// Basic parameters:
//---------------------
steps = 100;              // Steps per circular loop
blength = 0.6;            // Basic length of polygon
width = 0.3;              // Basic width of polygon
radius = 0.6;             // Radius of circular loop
revolutions = 1;          // Number of loops to run
```

Пример 6-2: анимация из Waltzing многоугольника (2/4)

- Матрица многоугольника определяет длину & ширину краев. Измените их различным значениям и прямоугольник изменится на другой размер.
- Обратите внимание на использование %inf постоянной, чтобы заполнить недостающие аргументы в plot2d ()
- h=gca() объявляет h в качестве ручки
- Ручка впервые использована для установки цвета фона

```
// Basic equations & definitions:
//-----------------------------------
t = linspace(0,revolutions*2*%pi,steps)';
x_axis = radius*cos(t);      // x-axis of circular loop
y_axis = radius*sin(t);      // y-axis of circular loop
polygon = [-blength/2  blength/2  blength/2  -blength/2;...
            -width/2  -width/2  width/2  width/2];
          // Defines corners of polygon

// Set scale for isometric plot:
//-------------------------------
plot2d(%inf,%inf,frameflag=3, rect=[-1,-1,1,1], axesflag=0)
  h = gca();
  xtitle('Waltzing polygon')
  h.background = 1;          // Set background to black
```

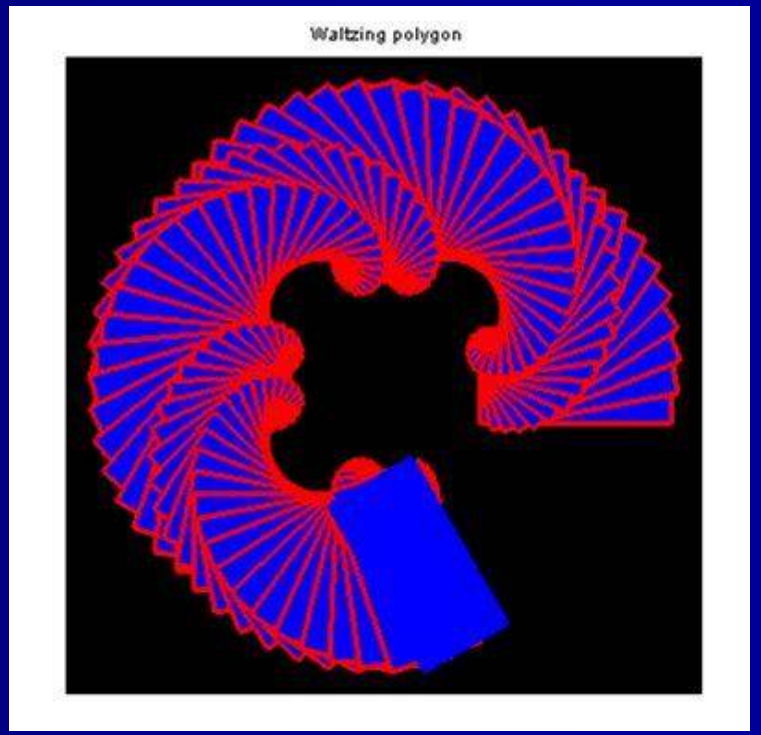
Пример 6-2: анимация из Waltzing многоугольника (3/4)

- Матрица выравнивает и получает многоугольник в новом угле. Изменение theta значения, чтобы увидеть некоторые интересные эффекты.
- Здесь ручка используется для установки цвета заливки фона; процесс выполняется xfpoly ()
- Следующая ручка определяет цвет границы; в этом случае функция выполнения является xpoly ()

```
// Plot rectangle as it waltzes its loop:
//---------------------------------------
turns = 3;                    // Number of turns per loop
for i=1:steps
  theta = turns*t(i);         // Angle of polygon alignment
  align = [cos(theta)  -sin(theta);...
           sin(theta)  cos(theta)]*polygon;
           // Realigns polygon
  h.foreground = 2;           // Set fill color to red
  xfpoly(align(1,:)+x_axis(i), align(2,:)+y_axis(i))
         // Fills polygon with defined color
  h.foreground = 5;           // Change to blue for border
  h.thickness = 3;            // Set border thickness to 3
  xpoly(align(1,:)+x_axis(i), align(2,:)+y_axis(i),'lines',1)
        // Draws polygon border in defined color
end
```

Пример 6-2: анимация из Waltzing многоугольника (4/4)

В этом скриншоте многоугольник (прямоугольник) сделавший чуть более трех четвертей своей часовой цикл. В то же время он развернулся 2 ¼ раза вокруг своей оси, и начался последний ¾ оборота. Есть 100 образцов позиции на полном цикле (шагов = 100;) и он будет завершен в течение нескольких секунд

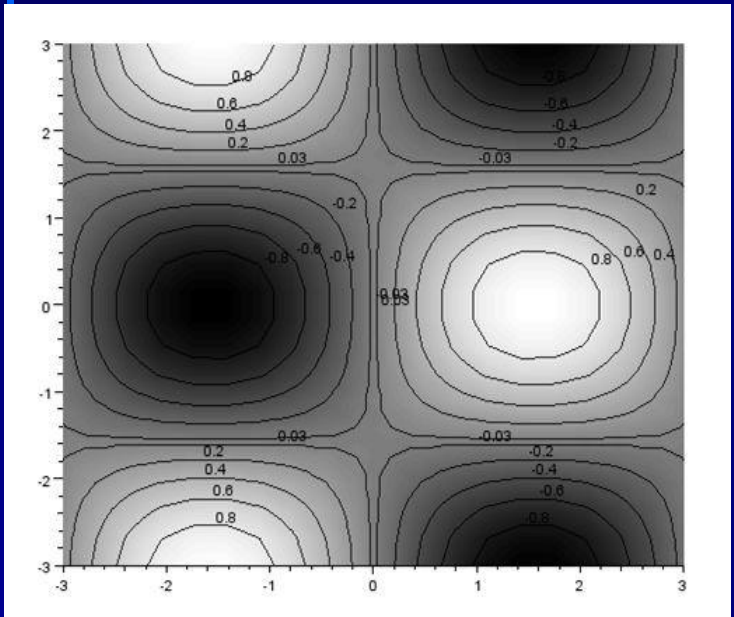


Пример 6-3 (1/2) : grayplot() & contour2d()

- Этот пример показывает, как серая цветовая гамма и контурные линии могут быть объединены, чтобы создать иллюзию 3D-пространства.
- LINSPACE () умножается на вектор 1x3, так как цветная карта ("третье измерение") должен быть mx3 матрица. Карта цветов может быть инвертирована с (1-LINSPACE ()) и нелинейная функция амплитуды могут быть добавлены с подчеркнуть эффектом.
- Функция Sgrayplot () сглаживает цвет участка по сравнению с основной функции grayplot ()
- Добавляются горизонтали

```
// grayplot_demo.sce /

// Gray area map with level curves using /
// grayplot()/Sgrayplot() & contour2d() to /
// create illusion of a 3D space /

clear,clc,clf();

// Color map definitions & initial declarations:
//-------------------------------------------------------------
f = gcf();
f.color_map = linspace(0,1,64)'*ones(1,3);
n = 20;                    // Plot resolution
x = linspace(-3,3,n);      // 3D plot limits
y = x;

// Plot function:
//-------------------
Z = sin(x)'*cos(y);        // Function to plot
Sgrayplot(x,y,Z)           // Smoothed grayplot

// Define and add level curves:
//-------------------------------------
level = [-.8 -.6 -.4 -.2 -.03 .03 .2 .4 .6 .8];
contour2d(x,y,Z,level);
```

Пример 6-3 (2/2): grayplot() & contour2d()



Влияние синусоиды и косинусоиды функций легко увидеть (обратите внимание, что происходит в центре графика)

Контурные линии становятся белыми, если цвет карты перевернуть

Шаги начинают проявляться в серой шкале если определение цветной карты изменить на LINSPACE (0,1,32), где аргумент 32 обозначает половину цветового разрешения

Изменив участок функции Sgrayplot () на grayplot (), вы увидите смысл переменной n = 20

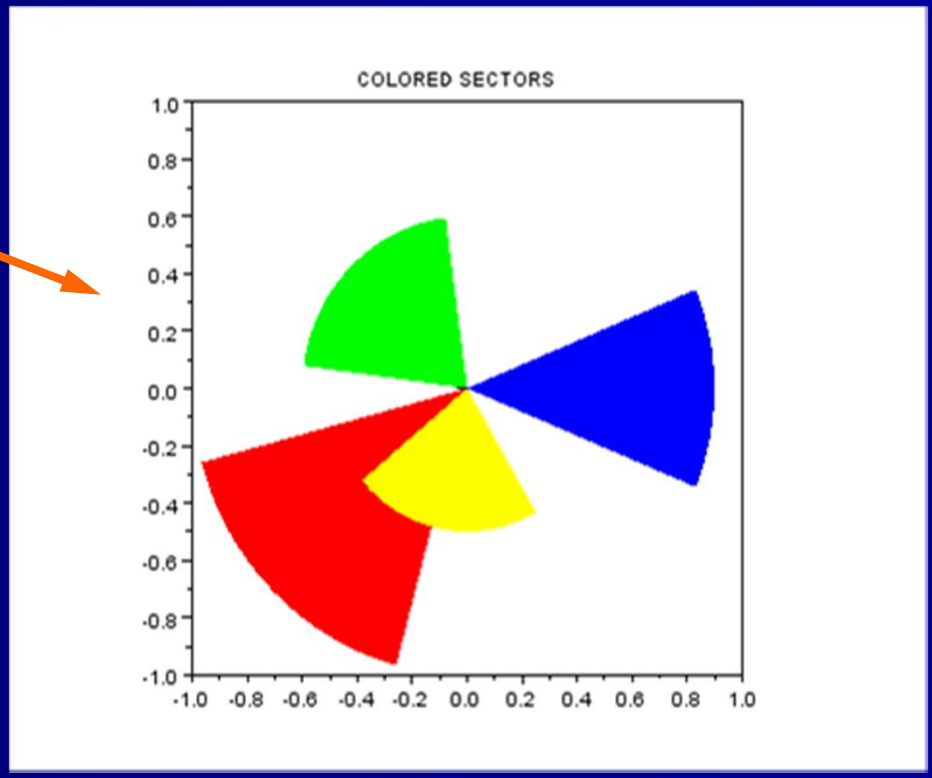
Пример 6-4: Сектор диаграммы, сценарий

- Этот сценарий основан на решении Пьера Ландо и показывает способ создания сектора диаграммы, в каждой отрасли, определив длину (радиус), направление, ширину и цвет.
- Решение можно рассматривать как более общем в случае использование функции pie() в Scilab, которую мы встретили в главе 9.
- Наиболее важной функцией в данном случае является xfarcs (), которую мы уже встречались в первой анимации демо (вектор дуги, конечно, также важен, так как он регулирует весь сюжет)

```
// create_sectors.sce

// Plots four colored sectors in predifined    /
// directions and with predifined widths       /

clear,clc,clf;

//          ---- SUBROUTINE ----               /
// The plot2d() function defines the figure,  /
// xfarcs() adds colored sectors to the plot  /

function create_sectors(r, angle, width, col)
    plot2d(%nan,%nan,-1,"031"," ",[-1,-1,1,1])
    arcs=[-r;r;2*r;2*r;(angle-width/2)*64;width*64];
    xfarcs(arcs,col)                  // Add sectors
    xtitle('COLORED SECTORS')
endfunction

//  ---- MAIN ---- /
// Define sectors:
//---------------------
rad = [.9,.6,1,.5]                 // Sector radii
angle = [0,135,225,270]            // Sector midpoints
width = [45,75,60,80]              // Sector widths
colors = [2,3,5,7]                 // Color definitions

// Call subroutine:
//---------------------
create_sectors(rad,angle,width,colors)

// ---- END MAIN ---- /
```

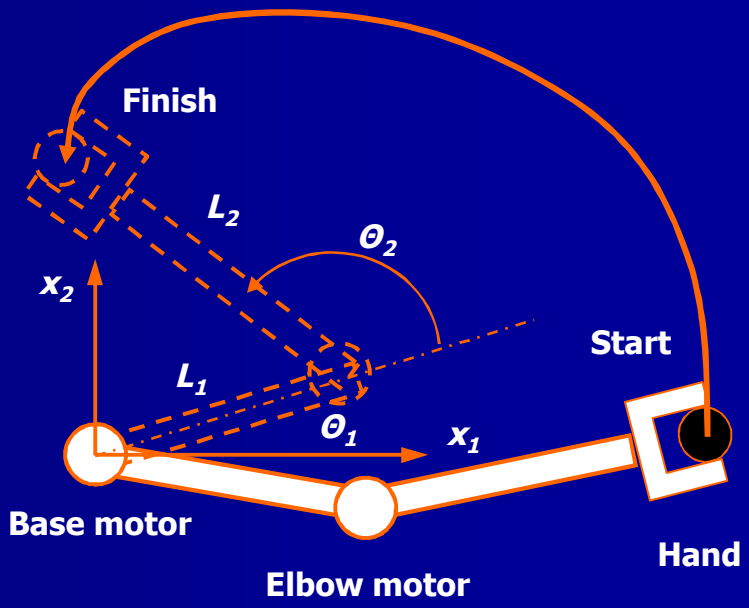
Пример 6-4: сектор диаграммы, график

- Вот график, хороший и красивый. Мы можем убрать оси, изменяя plot2d () Аргумент '031 ', на '030’
- Желтый сектор как раз находится на вершине (последний элемент в списке векторов). В практических приложениях, при выполнении автоматического построения некоторого процесса, мы должны были бы приложить больше усилий для получения сюжета, какой мы хотим



Пример 6-5: Робот (1/6)Введение

- Последние веб-дискуссии по относительным достоинствам Scilab, Matlab и Octave заставило меня по-новому взглянуть на ручное преобразования сценариев Matlab
- Этот случай с двумерным движущемся роботаом от Чепмен, SJ:. Matlab Программирование для инженеров, 2-е изд, (издатель и год неизвестен), стр. 202-206
- Дело демонстрирует практическое применение матриц. Смотрите Чепмен для всестороннего обсуждения
- Оригинальный сценарий можно найти на следующих слайдах; преобразованные сценарии с добавленными комментариями находятся на следующих двух слайдах



Пример 6-5: Робот (2/6), Matlab сценария

```
% Robot arm motion script
%
% Initial values, angles in degrees
tf = 2;
theta10 = -19*pi/180;
theta1tf = 43*pi/180;
theta20 = 44*pi/180;
theta2tf = 151*pi/180;
%
% Equations for a coefficients
T = [  tf^5      tf^4       tf^3
       5*tf^4   4*tf^3    3*tf^2
      20*tf^3  12*tf^2   6*tf ];
c = [ theta1tf-theta10; 0; 0 ];
disp('Coefficients for theta1 motion:')
a = T\c
%
% Equations for b coefficients
d = [ theta2tf-theta20; 0; 0 ];
disp('Coefficients for theta2 motion:')
```

```
b= T\d
%
% Equations of motion
L1 = 4;
L2 = 3;
t = linspace(0,2,401);
tq = [ t.^5; t.^4; t.^3 ];
theta1 = theta10 + a'*tq;
theta2 = theta20 + b'*tq;
x1 = L1*cos(theta1) + L2*cos(theta1 + theta2);
x2 = L1*sin(theta1) + L2*sin(theta1 + theta2);
%
% Plot path of hand
plot(x1,x2),...
xlabel('x_1'),...
ylabel('x_2'),...
    title('Path of robot hand'),...
    text(4.3,0,'t=0s: (x_1,x_2) = (6.5,0)'),...
    text(0.2,2,'t=2s: (x_1,x_2) = (0,2)')
```

Пример 6-5: Робот (3/6), преобразование Scilab (1/2)

Совместные двигатели управляются следующими полиномиальными выражениями

$$\Theta_1(t) = \Theta_1(0)+a_1t^5+a_2t^4+a_3t^3+ +a_4t^2+a_5t$$

$$\Theta_2(t) = \Theta_2(0)+b_1t^5+b_2t^4+b_3t^3+ +b_4t^2+b_5t$$

Матричные уравнения устанавливаются и решаются для коэффициента векторов (а, б), с использованием заданных начальные значения Θ (0) и конечное значения Θ (TF), и результаты используются для построения путей робота

```
// robot_motion.sce

// Robot arm motion in two dimensions using a fifth-degree /
// polynomial to control the motion. See Chapman, S.J.: /
// "Matlab programming for Engineers," 2nd ed., for a /
// detailed discussion. /

clear;clc,clf;

// Initial values, angles in degrees:
//---------------------------------------
tf = 2;                          // Finish time
theta10 = -19*%pi/180;           // Theta 1 start position
theta1tf = 43*%pi/180;           // Theta 1 final position
theta20 = 44*%pi/180;            // Theta 2 start position
theta2tf = 151*%pi/180;          // Theta 2 final position

// Equations for a coefficients (velocity
// constraints have been taken into account):
//-----------------------------------------------------
T = [    tf^5     tf^4     tf^3
       5*tf^4  4*tf^3  3*tf^2       // Angular velocity
     20*tf^3 12*tf^2  6*tf    ];    // Angular acceleration
c = [ theta1tf - theta10; 0; 0 ];   // Theta 1 movement
a = T\c                             // Coefficient vector a
disp(['Coefficients for theta1 motion:'])
disp([string(a')])
```

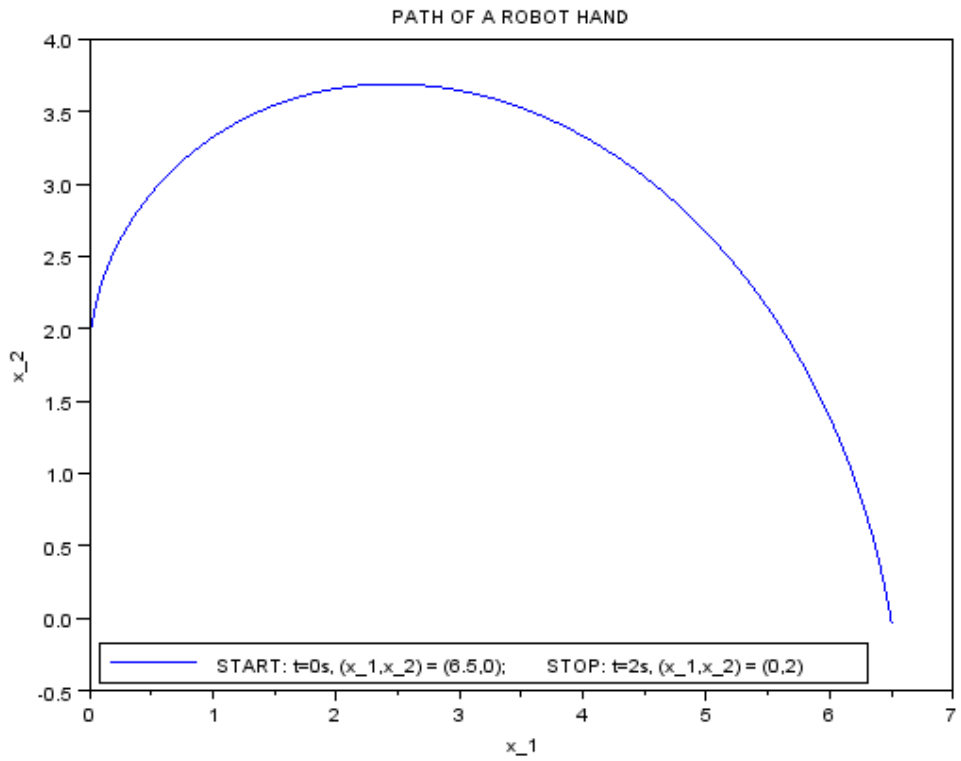
Пример 6-5: Робот (4/6), преобразование Scilab (2/2)

Требуя, что скорость и ускорение при T = 0 равным нулю, полиномиальные коэффициенты a5 и a4 обращаются в нуль. Это ограничивает размер T матрицы (предыдущий слайд) в 3x3

Вычисленный коэффициент векторов a и b используются для определения угловых скоростей, основание которых определяющихся в x1 и x2 координатах

```
// Equations for b coefficients:
//-------------------------------------
d = [ theta2tf - theta20; 0; 0 ];      // Theta 2 movement
b= T\d                                 // Coefficient vector b
disp(['Coefficients for theta2 motion:'])
disp([string(b')])

// Equations of motion:
//-----------------------------
L1 = 4;                                // Length of upper arm [feet]
L2 = 3;                                // Length of lower arm [feet]
t = linspace(0, 2, 401);               // Computation steps
tq = [ t.^5; t.^4; t.^3 ];
theta1 = theta10 + a'*tq;              // Base motor angular speed
theta2 = theta20 + b'*tq;              // Elbow motor angular speed
x1 = L1*cos(theta1) + L2*cos(theta1 + theta2);  // x1 position
x2 = L1*sin(theta1) + L2*sin(theta1 + theta2);  // x2 position

// Plot path of hand, add labels & legend:
//-------------------------------------------------
plot(x1,x2),..
   xlabel('x_1'),..
   ylabel('x_2'),..
   title('PATH OF A ROBOT HAND'),..
   h1 = legend(['START: t=0s, (x_1,x_2) = (6.5,0); ..
      STOP: t=2s, (x_1,x_2) = (0,2)'], 3)
```

Пример 6-5: Робот (5/6), график и дисплей



Coefficients for theta1 motion:
!0.2028945 -1.0144726 1.3526302 !

Coefficients for theta2 motion:

!0.3501567 -1.7507834 2.3343779 !

Пример 6-5: Робот (6/6), обсуждение

Руководство преобразования из сценариев Matlab в Scilab было достаточно простым. Произошедшие изменения:

- % Знак комментариев MATLAB пришлось изменить на / /
- Встроенную функцию pi MATLAB пришлось заменить на %pi
- Апострофы (кавычек) пришлось переписать, но только потому, что копия-вставка дает неправильный тип
- Команда DISP () должна быть изменена, потому что Scilab не выводит a и б коэффициенты, даже если соответствующие линии (a=T\d and b=T\d) конец без точки с запятой (ошибка?)
- команды MATLAB в `text()` не распознается Scilab (см. главу 9). Это позволяет легенда начала планируется разместить в точных точках
- Помощь Браузер не дает ответа, что делать
- Scilab-для-Matlab пользователей сборник по Beil & Grimm-Strėlė упоминает этот частный случай, но не предлагает решение
- Заключение: Мы должны придерживаться простых команд легенды Scilab

Пример 6-6: анимация с планетой и луной, интро

- Задача состоит в том, чтобы оживить планету с Луной вращающейся вокруг нее. Если это возможно, фигуры должны иметь разные цвета
- В задаче появляется первая трудность в поиске способа сохранить планету неподвижной, а луну заставить вращаться. Мое решение перекроить оба тела для каждого шага, что Луна движется. Slow, но это работает
- Вторая трудность заключается в предоставлении фигурам различных цветов. Команда color_map ручка хорошая, но она работает на рисунке уровнем и можно применить только один цвет. Представленное решение не идеально, так как только края границ, образующих сферы имеют разные цвета (это может быть сделано на уровне Entity)
- Третья проблема заключается с выравниванием коробки. Это будет обсуждаться на участке слайда

Экс 6-6: планета и Луна, сценарий (1/3)

- Сферы (планета, луна) построены из прямоугольных граней. Значения граней вычисляются здесь, в подпрограмме грани `facet()`
- Основные переменные для планеты

```
// planet_moon1.sce

// Animation with a moon rotating around a planet.  /
// The spheres are ccomposed of 3D X, Y, and Z      /
// facets using the surf() function to plot.        /

/ clear,clc,clf;

// **** SUBOUTINE **** //
// Attach defined points to the spheres:
function [x, y, z] = facet(v, h)
   x = cos(v)'*cos(h);              // Facet x-matrix
   y = cos(v)'*sin(h);              // Facet y-matrix
   z = sin(v)'*ones(h);             // Facet z-matrix
endfunction

// **** MAIN **** //
// Define planet & moon variables:
//-----------------------------------------
// Planet (p), 10x10 degree grid:
vp = linspace(-%pi/2,%pi/2,18);  // 18 steps vertically
hp = linspace(0,2*%pi,36);       // 36 steps horizontally
rp = 2;                          // Planet radius
```

Пример 6-6: планета и Луна, сценарий (2/3)

- Основные переменные для Луны, как для самого и его расположение в пространстве Луны.
- ПЕРЕЙТИ НА подпрограмму `facet()` для вычисления фасеточных матрицы
- Основные определения участка

```
// Moon (m), 20x20 degree grid & offset from origin:
vm = linspace(-%pi/2,%pi/2,9);    // 9 steps vertically
hm = linspace(0,2*%pi,18);        // 18 steps horizontally
rm = 0.3;                         // Moon radius
Rm = 2.1;                         // Moon offset
Az = 0;                           // Moon start point
n = 1;                            // # of moon revolutions
step = 100                        // # of steps/revolution

// Define facets for spheres using subroutine facet():
//-------------------------------------------------------------------
[Xp,Yp,Zp] = facet(vp,hp);        // Planet
[Xm,Ym,Zm] = facet(vm,hm);        // Moon

// Plot commands (box, planet, moon):
//-----------------------------------------------
// Define 3D box, put double buffer on, define surface:
a = gca();
  a.data_bounds = [-5,-5,-3; 5,5,3];    // 3D box size
f = gcf();
  f.pixmap = "on";                      // Double buffer
  f.color_map = hotcolormap(32);        // Surface color
```

Пример 6-6: планета и Луна, сценарий (3/3)

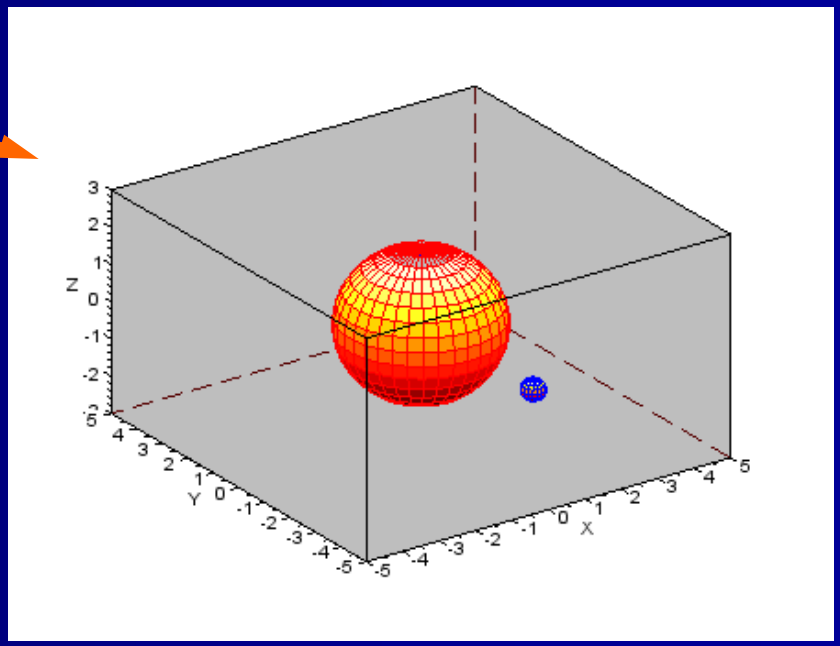
- Петля для начала вращения
- Удалите старые графики
- Положите цвет на коробку
- Нажмите данные планеты к первому буферу
- Пересчитать восход Местонахождение & передавать данные первого буфера
- show_pixmap () = толчок участока для экрана

```
// Plot planet & rotating moon:
for Az = 0 : 2*%pi/step : n*2*%pi

  // Delete previous entities (planet & moon):
  if (a.children~=[]) then      // IF frame contains graph...
    delete(a.children);         //      then delete graph
  end

  // Plot planet & define facet edges:
  a.background = color('grey');     // Box wall color
  surf(rp*Xp, rp*Yp, rp*Zp);        // Plot planet
    e1 = gce();
    e1.foreground = color('red');   // Facet edge color

  // Plot moon & define facet edges:
  x_loc = Rm*sin(Az);               // Location on x axis
  y_loc = Rm*cos(Az);               // Location on y axis
  C = Rm*[x_loc, -y_loc, 0]         // Moon center
  surf(C(1)+rm*Xm, C(2)+rm*Ym, C(3)+rm*Zm); // Plot moon
    e2 = gce();
    e2.foreground = color('blue');  // Facet edge color
  show_pixmap();
end
f.pixmap = "off";

// **** END MAIN **** //
```

Пример 6-6: анимация с планетой и луной, участок

А вот красота. Луна вращается против часовой стрелки и показана в ее исходном положение.

Как было сказано выше, эта задача не обошлась без проблем. Одна вещь, которую мне не удалось сделать, это наклонить окно несколько иначе. Ручка команда = GCA (); a.rotation_angles = [альфа, тета] просто отказались сотрудничать и углы остановились в 51 ° и -125 ° соответственно (окончательное ошибка.)



Измеряя с tick(); ... tock(), на выполнение каждого шага у луны занимает около 150 миллисекунд

Dr. EW
Johnny Heikell

20. До свидания!

Заключительные слова, чтобы сопровождать вас в вашей борьбе за выживание.

Вот и все!

- Мы достигли конца нашего путешествия. Дорога была длинная и более сложная чем я ожидал
- Существует гораздо больше всего, что есть Scilab но мы находимся на пути изучения этого материала (даже в офисных пакетах, вы не догадываетесь, как много всего в них есть, даже если вы используете его каждый день)
- Наиболее важным следующим шагом Scilab будет самостоятельное моделирования самостоятельно, решение проблемы в нашей конкретной сфере интересов.
- И ради всех, постоянно напоминать команде Scilab о необходимости всеобъемлющего учебника.
- Всего самого наилучшего!
  JH