scilab enterprises

Scilab для начинающих

Этот документ был написан совместно компанией Scilab Enterprises и Кристин Гомес, учителем математики в лицее Декарта в Антони, О-де-Сен, в 2013 году.
Он был обновлен Клодом Гомесом в январе 2025 года.

Оглавление

Введение

О данном документе 3
Установка Scilab 3
Список рассылки и информации 3
Дополнительные ресурсы 3

Глава 1 - Знакомство со Scilab

Общая среда и консоль 4
Простые численные расчеты 5
Строка меню 6
Редактор 7
Графическое окно 8
Онлайн-справка 9
Управление окнами и настройка рабочего пространства 10

Глава 2 - Программирование

Переменные, назначение и отображение 11
Циклы 14
Тесты 16
Графики в 2 и 3 измерениях 17
Дополнительная информация о матрицах и векторах 22
Некоторые полезные функции 26
Проблемы точности 28
Решение дифференциальных уравнений 29

Глава 3 - Полезные функции Scilab

Для анализа
31
Вероятность и статистика
31
Для отображения и построения графиков
32
Утилиты
32

Введение

Об этом документе

Цель этого документа - шаг за шагом рассказать о различных базовых функциях программы Scilab пользователю, который никогда раньше не пользовался программами для расчетов. В этом изложении намеренно ограничивается самое необходимое, чтобы облегчить освоение Scilab.

Расчеты, графики и иллюстрации выполнены в Scilab 2025.0.0. Поэтому вы можете воспроизвести все представленные команды из этой версии.

Установка Scilab

Scilab - это программа для численных расчетов, которую каждый может скачать бесплатно. Scilab доступен для Windows, Linux и MacOS и может быть загружен по следующему адресу: http://www.scilab.org/.

Рассылка и информационный список

Чтобы облегчить обмен мнениями между пользователями Scilab, были созданы списки рассылки (французский список, список для мира образования, международный список на английском языке). Принцип прост: подписчики могут общаться друг с другом по электронной почте (вопросы, ответы, обмен документами, обратная связь и т. д.).

Чтобы просмотреть доступные списки и зарегистрироваться, зайдите на сайт https://www.scilab.org/about/community/mailing-lists.

Дополнительные ресурсы

На сайте Scilab можно найти учебники по использованию Scilab по адресу https://www.scilab.org/tutorials.

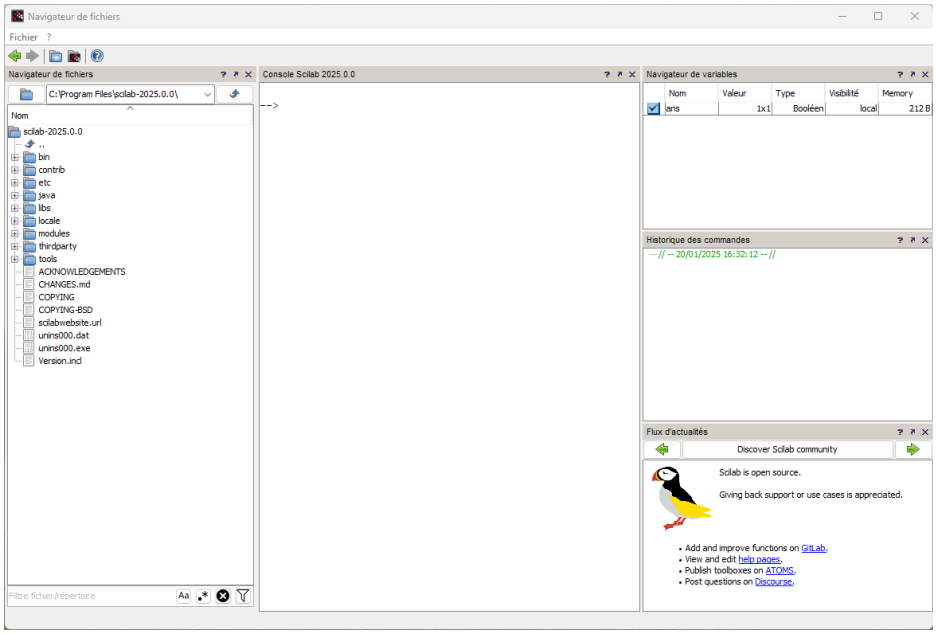
Глава 1 - Знакомство со Scilab

Полезное рабочее пространство в Scilab состоит из нескольких окон:

- Консоль для выполнения расчетов,
- Редактор для написания программ,
- Графические окна для отображения графики,
- Помогите.

Общая среда и консоль

После двойного щелчка на значке Scilab для запуска программы в среде Scilab по умолчанию отображаются следующие пристыкованные окна - консоль, браузеры файлов и переменных, история команд (см. раздел "Управление окнами и настройка рабочего пространства", стр. 10):



В консоли, после командной строки "**-->**", просто введите команду и нажмите Enter (Windows и Linux) или Return (MacOS) на клавиатуре, чтобы получить соответствующий результат.

```
--> 57/4
 годы =
    14.25
--> (2+9)^5
 годы =
    161051.
```

Обратите внимание
Напротив результата **ans** означает

Вы можете вернуться назад в любой момент, используя стрелки клавиатуры ← ↑ → ↓ или мышь, причем клавиши влево и вправо позволяют изменять команды, а клавиши вверх и вниз - вернуться к ранее выполненной команде.

Простые числовые вычисления

Все вычисления, выполняемые Scilab, являются численными. Scilab вычисляет с помощью матриц (см. главу 2, стр. 22).

+^Операции записываются как " " для сложения, "**–**" для вычитания, "*****" для умножения, "**/**" для деления и " " для экспоненты. Десятичная точка в десятичных числах обозначается полной остановкой. Например :

```
-->2+3.4
 годы =
    5.4
```

Для корректной работы вычислений важно учитывать регистр (верхний и нижний). Например, в команде **sqrt**, которая вычисляет квадратный корень :

```
-->sqrt(9)
 годы =
    3.
```

в то время как

```
--> SQRT(9)
Переменная не определена:
SQRT
```

Специальные номера

%e и **%pi** обозначают e и π соответственно:

```
--> %e
 %e =
    2.7182818
```

```
--> %pi
 %pi =
    3.1415927
```

%i представляет комплексную переменную **i** на входе и отображает **i** на выходе:

```
--> 2+3*%i
 годы =
    2. + 3.i
```

Чтобы не отображать результат

Если добавить точку с запятой ";" в конец командной строки, вычисления будут выполнены, но результат не будет отображен.

```
-->(1+sqrt(5))/2;
```

```
--> (1+sqrt(5))/2
 годы =
    1.6180340
```

Чтобы запомнить имя функции

Названия основных функций приведены в главе 3 данного документа (стр. 31). Например :

```
--> exp(10)/factorial(10)
 годы =
    0.0060699
```

Обратите внимание. Доступные функции также перечислены в разделе справки, доступ к которому

Вы можете использовать клавишу табуляции → на клавиатуре, чтобы завершить имя функции или переменной, для которой вы указали первые несколько букв.

Например, если вы напечатаете :

```
--> Факт
```

и нажмите клавишу табуляции, появится небольшое окно, в котором вы сможете выбрать одну из всех функций и имен переменных, начинающихся с **fact**, например **factorial** и **factor**. Просто дважды щелкните на нужной функции или выберите ее с помощью мыши или клавиш ↑ ↓ и нажмите Enter (Windows и Linux) или Return (MacOS), чтобы вставить ее.

Строка меню

В частности, вы будете использовать меню, перечисленные ниже.

Приложения

- История ордеров позволяет найти все ордера за предыдущие сессии и за текущую сессию.
- Браузер переменных позволяет найти все переменные, которые ранее использовались во время одной сессии.

Издание

Параметры (в меню **Scilab** на MacOS) позволяют устанавливать и настраивать цвета, шрифты и размеры шрифтов в консоли и в редакторе, что очень удобно при проецировании на экран.

Нажмите кнопку **Очистить консоль**, чтобы очистить все содержимое консоли. При этом история сохраняется, а расчеты, выполненные во время сеанса, остаются в памяти. Вы всегда можете вернуться к команде, которая была удалена, с помощью клавиш со стрелками на клавиатуре.

Управление

Чтобы прервать выполнение программы, вы можете :

- **Приостановите** работу программы или нажмите **Control > Pause** в строке меню (Ctrl X в Windows и Linux или Command X в MacOS), если программа уже запущена. Во всех случаях командная строка "**-->**" сменится на "**-1->**", затем на "**-2->**"... если операция будет повторена.
- Чтобы вернуться к моменту, когда программа была прервана, введите **resume** в консоли или нажмите **Control > Resume**.
- Чтобы окончательно остановить вычисление без возможности возврата, введите **abort** в консоли или нажмите **Control > Abort** в строке меню.

Издатель

Ввод текста непосредственно в консоль имеет два недостатка: невозможно сохранить, а если набрано несколько строк инструкций, внести изменения не так

просто. Чтобы связать несколько инструкций воедино, следует воспользоваться редактором.

Откройте редактор

 Чтобы открыть редактор из консоли, нажмите на первую иконку на панели инструментов или на **Applications > SciNotes** в строке меню.

Редактор откроется с файлом по умолчанию под названием "**`Untitled 1`**".

Писать в редакторе

Вы набираете текст в редакторе, как в любом текстовом процессоре.

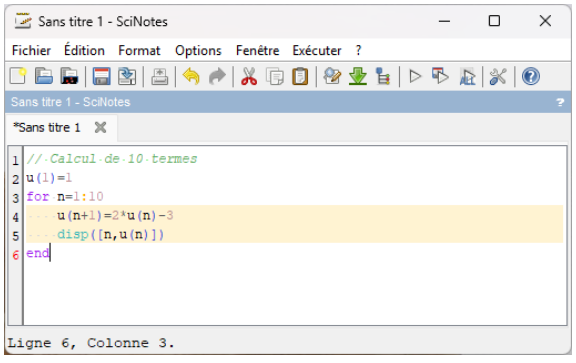
В текстовом редакторе открывающие и закрывающие скобки, команды конца цикла, функции и теста появляются автоматически. Однако вы можете отключить эти две функции в меню **Параметры > Автозавершение**, щелкнув по двум пунктам ниже, которые активированы по умолчанию:

- **(,[,...**
- **если, функция,...**

В принципе, вы должны переходить к строке после каждой инструкции, но можно набирать несколько инструкций на одной строке, разделяя их точкой с запятой "`;`".

При запуске цикла или теста автоматически применяется смещение начала строки, называемое отступом.

(u_n)В следующем примере мы вычисляем 10 членов последовательности, заданной : $\{u_1=1 u_{n+1}=2u_n-3$



Обратите внимание
- Чтобы написать комментарии, которые не будут учитываться в расчетах, перед ними поставьте "`//`".
- Чтобы изменить шрифт, нажмите **Options > Preferences**.
- Когда вы пишете программу, отступы делаются автоматически. Если это не так, нажмите **Format >**

Регистрация

Вы можете сохранить любой файл, нажав **Файл > Сохранить как**.

Расширение ".sce" в конце имени файла автоматически запустит Scilab при его

открытии (кроме Linux и MacOS).

Скопируйте в консоль, запустите программу

При нажатии на кнопку Run в строке меню вы получите три варианта:

- Выполнить "**...файл без эха**" (Ctrl Shift E в Windows и Linux, Cmd Shift E в MacOS): файл выполняется без записи программы в консоль (предварительно сохранив файл).
- Выполнить **"...файл с echo**" (Ctrl L в Windows и Linux, Cmd L в MacOS): переписывает файл в консоли и выполняет его.
- Выполнить "**...к курсору, с эхом**" (Ctrl E в Windows и Linux, Cmd E в MacOS): переписывает выбранное мышью выделение в консоль и выполняет его, или выполняет данные в файле до позиции курсора, заданной пользователем.

Вы также можете использовать классическое копирование/вставку.

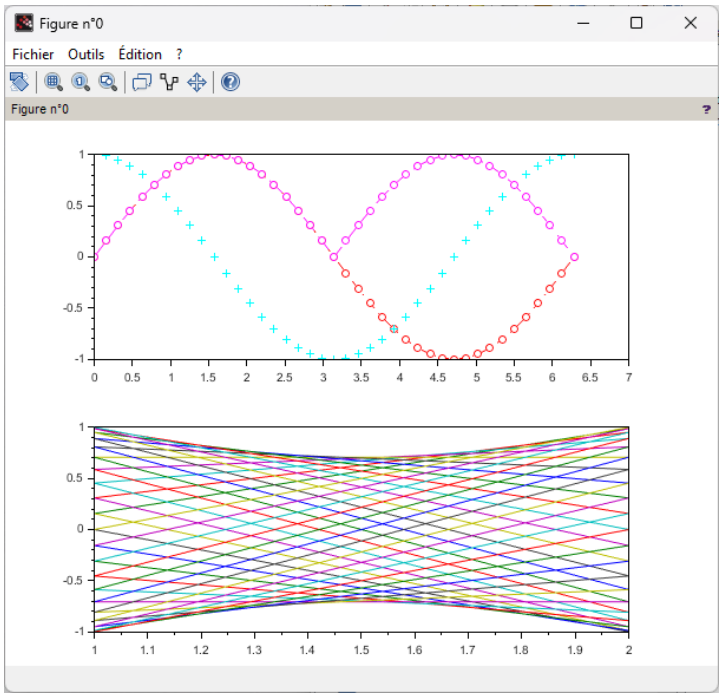
Графическое окно

Откройте графическое окно

При построении любого графика открывается графическое окно. Вы можете строить кривые, поверхности и облака точек (см. главу 2, стр. 17).

Пример кривой можно получить, набрав :

```
-->plot
```



Обратите внимание
- Чтобы удалить предыдущий след, введите
("четкая фигура" на английском языке).
- Чтобы открыть другое графическое окно, введите **`scf;`**
("установить текущий рисунок").
Если у вас открыто несколько графических окон, вы можете

Изменение маршрута

 Для увеличения масштаба используйте лупу. Чтобы увеличить масштаб в двух измерениях, нажмите на значок и с помощью мыши создайте прямоугольник, который будет формировать новый увеличенный вид. Чтобы увеличить масштаб в трех измерениях, щелкните на значке и создайте параллелепипед, который будет формировать новый увеличенный вид. Вы также можете увеличить масштаб с помощью колеса прокрутки на мыши.  Чтобы вернуться к исходному экрану, нажмите на другую лупу.

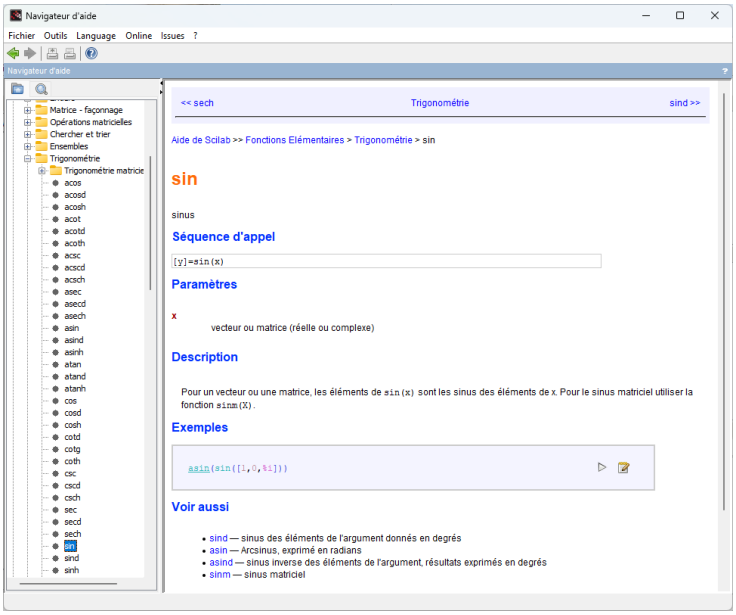
 Значок можно использовать для поворота фигуры (особенно полезно в 3D) с помощью действий правой кнопкой мыши, которые будут сопровождаться сообщениями в нижней части окна.

Для более точных изменений нажмите **Edit > Figure properties** или **Edit > Axis properties.** Откроется окно "**Графический редактор**", и вы сможете следовать инструкциям (эта опция пока недоступна на MacOS).

Онлайн-помощь

Чтобы получить доступ к интерактивной справке, нажмите на **? > Scilab Help** в строке меню или введите :

```
-->doc
```



Обратите внимание
Часть справки доступна на французском языке, остальная часть - на английском.
Примеры использования можно запускать в Scilab и редактировать в

Чтобы получить помощь по функциям, введите в консоль **doc** (ранее - **help**) и имя нужной функции. Например, введите :

```
>doc sin
```

отобразит функцию **sin** (синус).

Управление окнами и настройка рабочего пространства

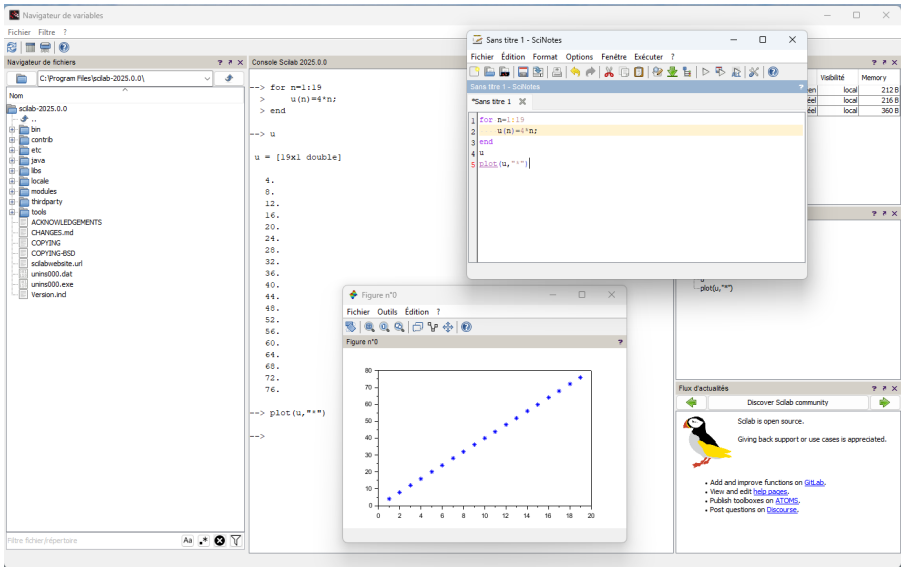
Как и в случае со средой Scilab по умолчанию, которая включает в себя окна

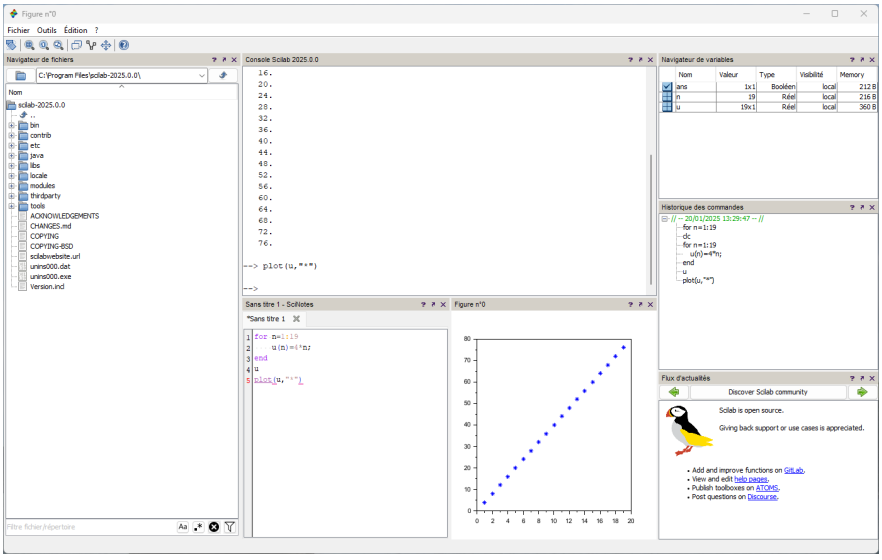
консоли, браузеры файлов и переменных и историю команд, все остальные окна Scilab могут быть перемещены в одном окне. Например, пользователь может выбрать размещение редактора в среде Scilab по умолчанию.

Чтобы поместить одно окно внутрь другого, сначала найдите синюю горизонтальную полосу в Windows или черную полосу в MacOS и Linux, расположенную в верхней части окна под панелью инструментов и содержащую вопросительный знак справа.

- В Windows и Linux щелкните на этой панели левой кнопкой мыши и, удерживая ее, переместите стрелку мыши на нужное окно.
- В MacOS щелкните на этой панели и, удерживая кнопку мыши, перетащите ее в нужное окно.

Появится прямоугольник, указывающий на будущее положение окна. Когда положение станет таким, как вы хотите, отпустите кнопку мыши. Чтобы отменить и раскрыть окно, нажмите на маленькую стрелку справа от той же полосы.





Глава 2 - Программирование

В примерах, приведенных в этом документе, любая строка, которой предшествует "-->", является командой; остальные строки - это возвраты (результаты вычислений, сообщения об ошибках и т. д.). Не пишите "-->" в редакторе. Мы ввели его только для того, чтобы отличать командные строки от возвратов вычислений, которые отображаются в консоли после операции копирования/вставки. Представленные в таблице (без "-->" и без возвратов вычислений) команды отображаются так, как они были бы набраны в редакторе.

Переменные, назначение и отображение

Переменные

Scilab не является программой для формальных вычислений. Он вычисляет только с помощью чисел. На самом деле все вычисления производятся с помощью матриц, но это может остаться незамеченным. Даже если понятие матрицы нам неизвестно, мы используем векторы и последовательности чисел, которые на самом деле являются матрицами $1 \times n$ или $n \times 1$, так же как число является матрицей размерности 1×1.

Переменные не нужно объявлять заранее, но каждая переменная должна иметь значение. Например, запрос значения переменной **a** без предварительного присвоения ей значения приводит к ошибке :

```
-->a
Неопределенная переменная: a
```

Если присвоить переменной **a** значение, то ошибок больше не будет:

```
--> a=%pi/4
 a =
    0.7853982
--> a
 a =
    0.7853982
```

Можно использовать любое имя переменной, которое еще не определено системой:

```
--> Pisur2=%pi/2
 Pisur2 =
    1.5707963
```

Обратите внимание.
Как и в функциях Scilab, имена переменных не должны содержать подчеркиваний или

Если результат вычисления не присваивается переменной, значение автоматически присваивается переменной **ans** (ответ):

```
-->3*(4-2)
 годы =
    6.


--> лет
 годы =
    6.
```

Функции

Функции - это самый простой и естественный способ выполнения вычислений с использованием переменных и получения из них результатов.

Определение функции начинается с **function** и заканчивается **endfunction**. Например, чтобы преобразовать евро (e) в доллары (d) с обменным курсом (t), мы определяем функцию **dollars**. Переменными являются **e** и **t**, а образом - **d**.

```
-->функция d=dollars(e,t); d=e*t; endfunction


-->dollars(200,1.04)
 годы =
    208.
```

Чаще всего используются числовые функции с одной переменной. Например, две функции **f** и **g** задаются с помощью команд, приведенных ниже:

```
-->функция y=f(x); y=36/(8+exp(-x)); endfunction


-->функция y=g(x); y=4*x/9+4; endfunction
```

Обратите внимание
Переменные **x** и **y** являются фиктивными переменными, поэтому их имена могут быть

После того как функции определены, их можно использовать для вычисления значений:

```
--> f(10)
 годы =
    4.4999745
--> g(12.5)
 годы =
    9.5555556
```

Запрос на присвоение значения

=Назначение простое, с помощью символа " ".

Дисплей

Пишите на

При вводе имени переменной отображается ее значение, за исключением ";" в конце команды.

Крючки

Квадратные скобки используются для определения матриц (см. стр. 22). Как уже говорилось, вычисление матриц является основой всех вычислений, выполняемых в Scilab. Пробел или запятая используются для перехода от одного столбца к другому, а точка с запятой - для перехода от одной строки к другой.

Чтобы определить вектор-столбец и получить отображение столбца :

```
-->v=[3;-2;5]
 v = [3x1 double]
    3.
  - 2.
    5.
```

Для определения линейного вектора и получения линейного отображения :

```
-->v=[3 -2 5]
 v = [1x3 double]
    3.  - 2. 5.
```

Обратите внимание
Также можно ввести эту команду в виде: **v=[3,–**

Чтобы задать матрицу и вывести таблицу, выполните следующие действия:

```
-->m=[1 2 3;4 5 6;7 8 9]
 m = [3x3 double]
    1. 2. 3.
    4. 5. 6.
    7. 8. 9.
```

Обратите внимание
Вы также можете ввести эту команду в виде:
m=[1,2,3;4,5,6;7,8,9]

Диспансер

За функцией **disp** всегда следуют круглые скобки.

С предыдущим вектором **v** :

```
-->v(2)
 годы =
  - 2.


-->disp(v(2))
  - 2.
```

Чтобы отобразить строку символов (обычно предложение), заключите ее в перевернутые запятые :

```
-->disp("Боб выиграл")
  Боб выиграл
```

+Чтобы смешать слова и значения, используйте команду **string**, которая преобразует значения в символы, и " " между различными частями:

```
-->d=500;


-->disp("Боб выиграл "+string(d)+"dollars")
 Боб выиграл 500 долларов
```

Если предложение содержит апостроф, то для корректного отображения он должен быть удвоен в строке символов:

```
-->disp("Все верно")
 Просто
```

Петли

Увеличение

Оператор ":" используется для определения векторов чисел, координаты которых находятся в арифметической последовательности. Мы задаем "первое значение: шаг: последнее значение" (не обязательно достигнутое). Если шаг не указан, то по умолчанию он равен 1.

Например, чтобы определить вектор строк целых чисел, которые увеличиваются на 1 в диапазоне от 3 до 10:

```
-->3:10
 ans = [1x8 double]
    3. 4. 5. 6. 7. 8. 9. 10.
```

или увеличить на 2 от 1 до 10:

```
-->1:2:10
```

```
 ans = [1x5 double]
    1. 3. 5. 7. 9.
```

или уменьшение на 4 между 20 и 2:

```
-->u=20:-4:2
 u = [1x5 double]
    20. 16. 12. 8. 4.
```

Для

Простейшая структура цикла для известного числа итераций - **`for ... end`**, что означает "Для ... конец for".

Пример: Вычисление 20 членов последовательности, заданной рекуррентным методом как : $\{u_1=4\, u_{n+1}=u_n+2n+3$

Алгоритм	Редактор Scilab

Подставьте 4 в u(1)
Для n от 1 до 20
u(n+1) принимает значение u(n) +2n+3
Отобразите n и u(n)
Конец для
	`u(1)=4;`
	`для n=1:20`
	`  u(n+1)=u(n)+2*n+3;`
	`  disp([n,u(n)])`
	`конец`

В то время как

Если вы хотите, чтобы цикл останавливался при достижении заданной цели, используйте форму **`while ... end`**, что означает "до тех пор, пока ... end of as long as".

Пример: в 2025 году я пересадил елку высотой 1,20 м. Она растет на 30 см в год. Я решил срубить ее, когда она превысит 7 м. В каком году я срублю елку?

Алгоритм	Редактор Scilab

Добавьте 1,2 к h (h = высота дерева).
Положите 2025 год в a (a = год)
До тех пор, пока h<7
h принимает значение h+0,3 (мое дерево растет)
a принимает значение a+1 (проходит один год)
	`h=1.2;`
	`a=2025;`
	`пока h<7`
	`  h=h+0,3;`
	`  a=a+1;`
	`конец`
	`disp("Я разрежу свое дерево на...`
	`+string(a))`

Конец до тех пор, пока
Показать a (за последний год)

Обратите внимание

Если команда слишком длинная для записи в одной строке, редактор автоматически переходит на следующую строку. Вы также можете поставить " .. (двоеточие) перед переходом на следующую строку

Тесты

Операторы сравнения

Сравнение чисел или проверка истинности или ложности утверждения - полезные тесты. Вот соответствующие команды:

Равный	Разное	Нижний	Превосходный	Меньше или равно	Больше или равно
==	**<>**	**<**	**>**	**<=**	**=>**
Правда	Ложь	И	Или	Нет	
%T	**%F**	**&**	**|**	**~**	

Обратите внимание

Будьте внимательны к точности. **==**Поскольку вычисления являются приблизительными, тест " " иногда дает неправильные ответы (см. Проблемы точности, стр. 28).

==<>Когда вы хотите сравнить два вектора (или две матрицы), тесты " " и " " сравнивают терм за термом. Например:

```
-->X=[1,2,5]; Y=[5,3,5];

-->X==Y
 ans = [1x3 boolean]
  F F T
```

Чтобы проверить, равны ли два вектора, мы используем **isequal**, а если они разные, то **~isequal**:

```
-->isequal(X,Y)
 годы =
  F

-->~isequal(X,Y)
 годы =
  T
```

Если... то

Классические структуры выглядят следующим образом:

- **`if ... then ... else ... end`** ("Если...then...else...end of if"),
- **`if ... then ... elseif ... then ... else ... end`** ("Если ... то ... или if ... то ... или ... конец if").

`if ... then` должно быть написано в той же строке, **`что и elseif ... then`**.

Пример: Алиса бросает три игральные кости.

- Если ей выпадут три шестерки, она выиграет €20,
- Если она получит три одинаковых результата, отличных от 6, то выиграет €10,
- Если она получит два одинаковых результата, то выиграет €5,
- В противном случае он ничего не зарабатывает.

Мы можем смоделировать бросок и рассчитать выигрыш Алисы, используя функции :

- **`большой`** (см. стр. 21),
- **`unique(D)`**, который сохраняет значения, встречающиеся в **`D`** несколько раз, только один раз,
- **`length(unique(D))`**, который дает размер полученного вектора, т.е. 1, если три термина равны, 2, если два термина равны.

Алгоритм	Редактор Scilab

Подставьте три значения в D
Если Алиса получила три шестерки, то
Алиса выигрывает 20 евро
Иначе, если он бросит 3 одинаковых кубика, то
Алиса выигрывает 10 евро
Иначе, если он бросает 2 одинаковых кубика, то
Алиса выигрывает 5 евро
В противном случае
Алиса ничего не выигрывает
Конец if
	`D=grand(1,3,"uin",1,6);`
	`если D==[6,6,6], то`
	`  G=20;`
	`elseif length(unique(D))==1 then`
	`  G=10;`
	`elseif length (unique(D))==2 then`
	`  G=5;`
	`else`
	`  G=0;`
	`конец`
	`disp("Алиса выиграла "+...`
	`string(G)+ " евро")`

Показать выигрыш Алисы

Чертежи в 2 и 3 измерениях

Графики на плоскости создаются с помощью команды **plot**. Вы можете выбрать цвет и внешний вид, заключив цвет и стиль точек в перевернутые запятые:

- Цвета

"b" = синий (по умолчанию), **"k"** = черный, **"r"** = красный, **"g"** = зеленый, **"c"** = голубой, **"m"** = пурпурный,

"y" = желтый, **"w"** = белый.

- Стили стежков

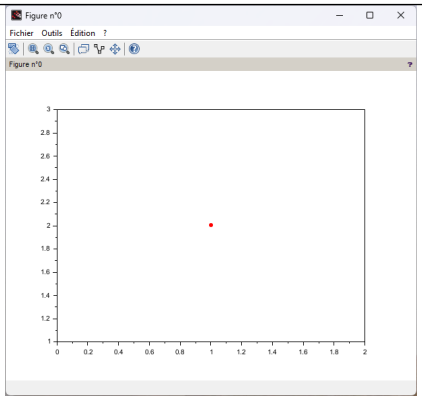
Ссылка (по умолчанию), или **"."**, **"+"**, **"o"**, **"x"**, **"*"**.

Другие опции доступны через : **"s"**, **"d"**, **"v"**, **"<"** и **">"**.

Основные макеты

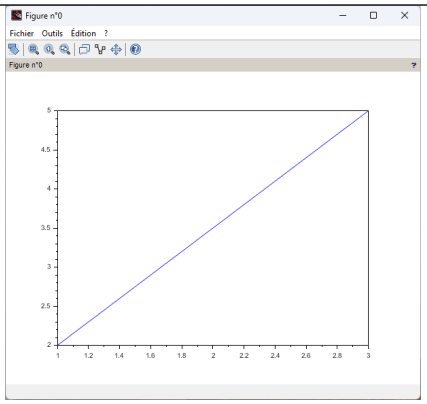
Чтобы нарисовать точку

Пометьте точку A(1; 2) красной точкой.

Редактор Scilab	Графическое окно
`plot(1,2,".r")`	

Чтобы нарисовать сегмент

Нарисуйте отрезок [AB] синим цветом (по умолчанию) с координатами A(1; 2) и B(3; 5).

Редактор Scilab	Графическое окно
`plot([1,3],[2,5])`	

Постройте кривые на плоскости, заданные функциями y=f(x)

$x \to f(x)xx$Для функции x мы сначала задаем значения x с помощью команды **`linspace`**, написав: **`x=linspace(a,b,n);`** где **`a`** - наименьшее значение переменной, **`b`** - наибольшее значение переменной, а **`n`** - количество значений, которые будут вычислены между **`a`** и **`b`**.

xНе забудьте про "**`;`**", иначе будет выведено n значений.

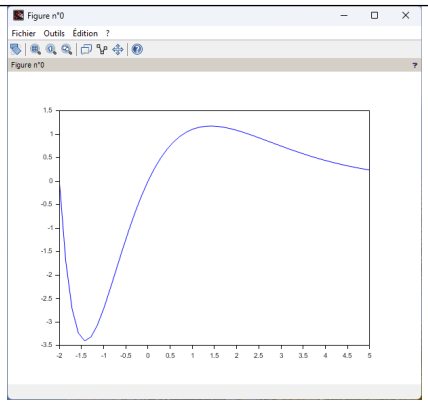
$f\,g$Например, пусть и - две функции, определенные на [-2; 5] через :

$$f(x)=(x^2+2x)e^{-x} \text{ и } g(x)=\sin\left(\frac{x}{2}\right)$$

f В программе ниже показана кривая для , которая по умолчанию отображается синим цветом.

Редактор Scilab	Графическое окно

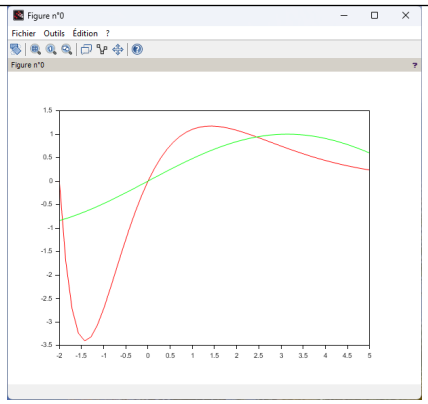
```
функция y=f(x)
  y=(x^2+2*x)*exp(-x)
endfunction
x=linspace(-2,5,50);
plot(x,f)
```



Добавив приведенную ниже программу, мы получим график двух кривых: красной - **`f`** и зеленой - **`g`**. Предыдущий график был очищен с помощью команды **`clf`**.

Редактор Scilab	Графическое окно

```
функция y=g(x)
  y=sin(x/2)
endfunction
x=linspace(-2,5,50);
clf
plot(x,f, "r",x,g, "g")
```



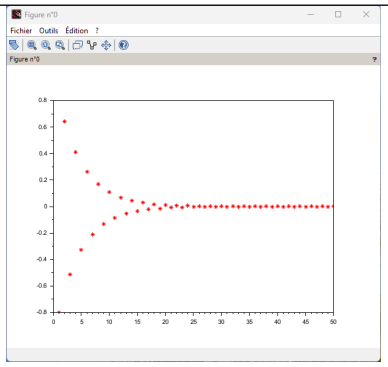
Построение облаков точек

Условия последовательности

$M(n,u(n))u(n)u$Чаще всего требуется построить график точек после вычисления координат вектора . Мы пишем **`plot(u, "*r")`**, указывая форму и цвет точек в облаке между инвертированными запятыми. Здесь мы выбрали красные звезды, которые не связаны между собой. По умолчанию точки синие и соединенные.

Редактор Scilab	Графическое окно

```
для n=1:50
  u(n)=(-0.8)^n;
конец
clf; plot(u, "*r")
```

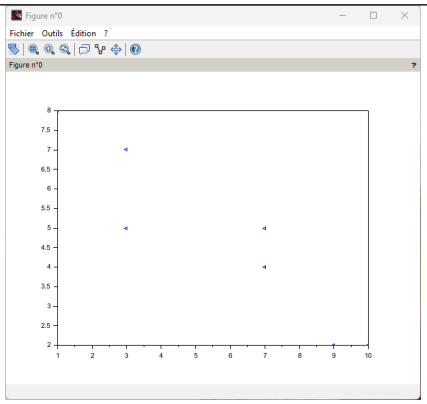


Двойная статистика

$M(X_i;Y_i)$Статистические облака задаются в виде двух векторов: назовем их X и Y, затем напишем **`plot(X,Y,"<")`**, чтобы изобразить облако точек синими треугольниками.

Редактор Scilab	Графическое окно

```
X=[1,3,3,7,7,9,10];
Y=[8,7,5,5,4,2,2];
clf; plot(X,Y,"<")
```



Трехмерные сюжеты

Scilab позволяет рисовать поверхности и кривые в пространстве, с большим количеством опций для работы со скрытыми гранями, цветами граней, точками обзора и т.д. Здесь мы приведем только два примера.

Функция **`surf`** используется для построения поверхности. $mn(Ox)(Oy)n\times mz_{ij}$ x_iy_jЭта функция принимает три входные переменные: **`x`**, **`y`** и **`z`**. **`x`** и **`y`** - векторы соответствующих размеров, соответствующие точкам на осях и . **`z`** - матрица размерности, элемент которой - размерность точки на поверхности с абсциссой и ординатой.

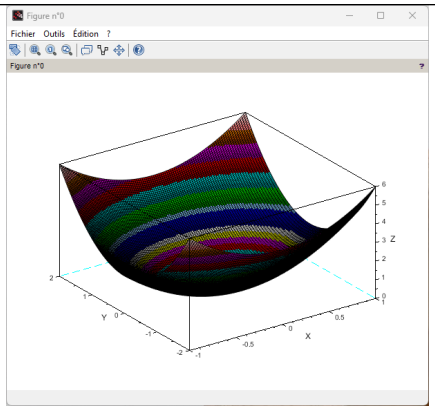
$z=f(x,y)$Чтобы построить поверхность, заданную функцией типа , нам нужно :

- Определите функцию f
- Вычислите **`z=feval(x,y,f)'`**.
  $m\times nijf(x_i,y_j)$**`feval(x,y,f)`** возвращает матрицу, элемент которой мы транспонируем с помощью апострофа " **`'`** ".
- Примените **`surf(x,y,z)`**.

$z = 2x^2 + y^2$Построение поверхности (эллиптического параболоида):

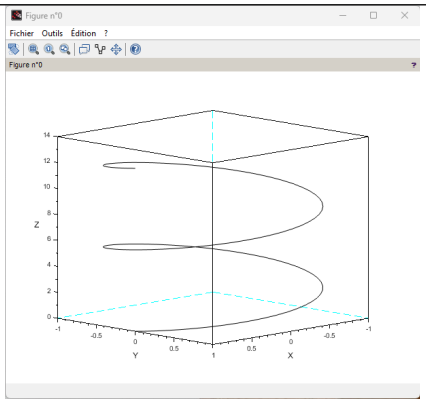
Редактор Scilab	Графическое окно

```
функция z=f(x,y)
  z=2*x^2+y^2;
endfunction
x=linspace(-1,1,100);
y=linspace(-2,2,200);
z=feval(x,y,f)';
clf
surf(x,y,z)
```



(x_i, y_i, z_i)Функция **param3d** используется для построения кривой в пространстве. **param3d** принимает три аргумента, **x**, **y** и **z**, которые представляют собой векторы одинакового размера, соответствующие точкам на кривой.

$(x = \cos\cos(t), y = \sin\sin(t), z = t)$Путь спирали, определяемый :

Редактор Scilab	Графическое окно
`t=linspace(0,4*%pi,100);` `clf ; param3d(cos(t),sin(t),t)`	

Моделирование и статистика

Широкий набор функций позволяет быстро и эффективно выполнять моделирование.

Случайные розыгрыши с заказом и скидкой

- $pmnpmnm \leq n$**grand(1,p, "uin",m,n)** возвращает вектор случайных целочисленных жребиев, взятых между и с целым положительным числом, и целыми числами и .

```
-->t= grand(1,4, "uin",1,6)
 t = [1x4 double]
    3. 1. 3. 6.
```

- $pabpaba \leq b$**grand(1,p, "unf",a,b)** возвращает вектор случайных вещественных жребиев, взятых между и с целым положительным числом, и вещественными и .

```
-->tr= grand(1,2, "unf",-1,1)
 tr = [1x2 double]
  - 0.7460264 0.9377355
```

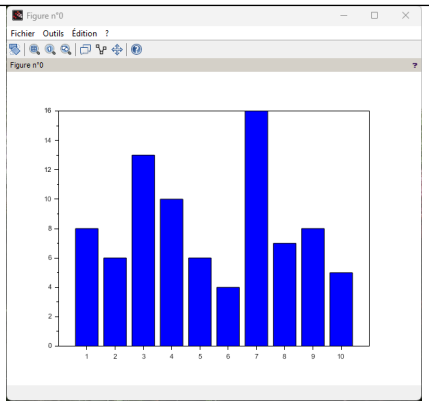
Статистика

Все обычные статистические функции перечислены на странице 31.

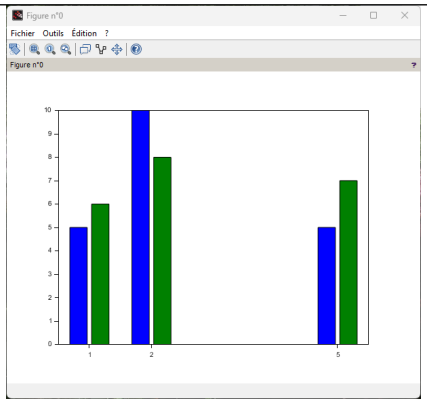
Помните об этом, в частности:

- Функция **bar(x,n,color)**, которая строит столбчатые диаграммы:

Редактор Scilab	Графическое окно

`x=[1:10];`
`n=[8,6,13,10,6,4,16,7,8,5];`
`clf; bar(x,n)`
	

- Для гистограммы, представляющей два ряда рядом друг с другом: ряд значений X и два ряда чисел n1 и n2. Для графика n1 и n2 должны быть векторами-столбцами, поэтому в примере ниже мы берем транспозиции :

Редактор Scilab	Графическое окно
`X=[1,2,5];n1=[5,10,5];n2=[6,8,7];` `clf; bar(X,[n1',n2'])`	

Необязательный аргумент **`color`** задает цвет, как в функции **`plot`**.

Дополнительная информация о матрицах и векторах

Доступ к элементам

Квадратные скобки используются для определения матрицы. Пробел или запятая используются для перехода от одного столбца к другому, а точка с запятой - для перехода от одной строки к другой.

```
-->m=[1 2 3;4 5 6]
 m = [2x3 double]
    1. 2. 3.
    4. 5. 6.
```

Обратите внимание
Также можно ввести эту команду в виде:
`m=[1,2,3;4,5,6]`

Кронштейны используются для доступа к элементам или их изменения.

```
-->m(2,3)
 годы =
```

```
    6.

-->m(2,3)=23
 m = [2x3 double]
    1. 2. 3.
    4. 5. 23.
```

Оператор ":" используется для обозначения всех строк или столбцов матрицы.

Чтобы получить вторую строку матрицы **m**, введите :

```
-->m(2,:)
 ans = [1x3 double]
    4. 5. 23.
```

и третий столбец :

```
-->m(:,3)
 ans = [2x1 double]
    3.
    23.
```

Чтобы получить транспонирование матрицы или вектора, используйте апостроф " **'** ":

```
-->m'
 ans = [3x2 double]
    1. 4.
    2. 5.
    3. 23.
```

Операции

Операции "*****" и "**/**" являются матричными операциями. Чтобы выполнить поэлементные операции, перед знаком операции поставьте точку: "**.***", "**./**".

```
-->A=[1,2,3;4,5,6]
 A = [2x3 двойных]
    1. 2. 3.
    4. 5. 6.
```

```
-->B=[1;1;2]
 B = [3x1 double]
    1.
    1.
    2.
```

```
-->A*B                                  Умножение матриц
```

```
 ans = [2x1 double]
    9.
    21.
```

```
-->A*A
    !--error 10
Оператор *: Неправильные размеры
для операции [2x3] * [2x3].
```

Размеры не соответствуют действительности

```
-->A.*A
 ans = [2x3 double]
    1. 4. 9.
    16. 25. 36.
```

Умножение элемента на элемент

```
-->2*(A+2)
 ans = [2x3 double]
    6. 8. 10.
    12. 14. 16.
```

Операция выполняется над каждым элементом, потому что 2 - это число

Дайте матрицу X такую, что X*A = A
Правильный ответ: :
1. 0
0 1.

```
-->A/A
 ans = [2x2 double]
    1. 5.061D-17
    -2.702-15 1.
```

Для точности расчетов результат может немного отличаться в зависимости от версии Scilab и операционной системы (см. подробности вычислений, стр. 28).

```
-->A./A
 ans = [2x3 double]
    1. 1. 1.
    1. 1. 1.
```

Дает матрицу, разделенную элемент на элемент

В случае с векторами :

```
-->C=1:4
 C = [1x4 double]
    1. 2. 3. 4.
```

```
-->C*C
Оператор *: Неправильные размеры
для операции [1x4] * [1x4].
```

Размеры не соответствуют действительности

```
-->C.*C
 ans = [1x4 double]
    1. 4. 9. 16.
```

```
-->(1)./C
 ans = [1x4 double]
    1. 0.5 0.3333333 0.25
```

Реверс элемент за элементом
Скобки вокруг 1 нужны для того, чтобы полная точка не считалась запятой, составляющей часть числа 1. Вы также можете написать **1 ./C** с пробелом между **1** и " **.**

Системные разрешения

Чтобы решить линейную систему AX = Y, где A - квадратная матрица, используйте обратную косую черту "\".

X = A \ Y.

Заметим, что операция Y / A даст (при условии правильной размерности) другой результат - матрицу Z такую, что Z A = Y.

Для решения системы : $\begin{pmatrix}1&2&3\\4&5&6\end{pmatrix} \times X = \begin{pmatrix}1\\1\end{pmatrix}$

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

-->Y=[1;1];

-->X=A\Y
 X = [3x1 double]
  - 0.5000000
    0.
    0.5000000

-->A*X
 ans = [2x1 double]
    1.000000
    1.
```

Некоторые полезные функции

Сортировать

Функция **gsort** используется для упорядочивания элементов вектора по возрастанию или убыванию.

```
-->v=[2,6,9,6,-4,0,2]
 v = [1x7 double]
    2. 6. 9. 6.  - 4. 0. 2.

--> gsort(v, "g", "i")
 ans = [1x7 double]
  - 4. 0. 2. 2. 6. 6. 9.

--> gsort(v)
 ans = [1x7 double]
    9. 6. 6. 2. 2. 0. - 4.
```

Размер

Функция **length** возвращает количество координат вектора. Функция **size** возвращает размерность (строки, столбцы) матрицы.

```
-->U=[1:10]
 U = [1x10 double]
    1. 2. 3. 4. 5. 6. 7. 8. 9. 10.

-->length(U)
 годы =
    10.

-->m=[1 2 3;4 5 6];

-->size(m)
 ans = [1x2 double]
    2. 3.
```

Сумма и произведение

Функции **sum** и **prod** вычисляют соответственно сумму и произведение элементов в своем аргументе.

```
-->U=[1:10];

-->sum(U)
 годы =
```

```
    55.


-->prod(U)
 годы =
    3628800.
```

Уникальный

Функция **unique** сохраняет элементы в векторе только один раз (даже если они повторяются несколько раз) и упорядочивает их по возрастанию. Она может быть очень полезна для тестирования.

```
-->v=[2,6,9,6,-4,0,2]
 v = [1x7 double]
    2. 6. 9. 6.  - 4. 0. 2.


-->unique(v)
 ans = [1x5 double]
  - 4. 0. 2. 6. 9.
```

Найти

Функция **find** ищет элементы в векторе или матрице и возвращает вектор, содержащий соответствующие индексы.

wНайти все элементы вектора, строго меньшие 5:

```
-->w=[1,5,3,8,14,7,3,2,12,6]; find(w<5)
 ans = [1x4 double]
    1. 3. 7. 8.
```

$w_1, w_3, w_7 w_8$Вектор результатов (1,3,7,8) говорит нам о том, что элементы и меньше 5.

wНайти все элементы вектора, равного 3:

```
-->w=[1,5,3,8,14,7,3,2,12,6]; find(w==3)
 ans = [1x2 double]
    3. 7.
```

$w_3 w_7$Вектор результатов (3.7) показывает, что элементы и равны 3.

Проблемы с точностью

Для расчета

Абсолютные значения чисел находятся в диапазоне примерно от $2{,}2\times10^{-308}$ до $1{,}8\times10^{+308}$.

Число **%eps**, равное **2.220446049D-16**, дает наименьшую относительную точность, которую можно ожидать при вычислениях, т.е. около 16 цифр.

Пример 1: Вычисление sin(π)

```
-->sin(%pi)
 годы =
1.225D-16
```

$\sin(\pi)$Значение выше не равно 0, но считается нулевым. На самом деле, по сравнению с максимальным значением функции синуса (т.е. 1), оно равно 0 с погрешностью менее **%eps**.

√❑Пример 2: Проверим, является ли треугольник со сторонами , 1 и 2 прямоугольным:

```
-->a=sqrt(3)
 a =
    1.7320508
```

```
-->b=1
 b =
    1.
```

```
-->c=2
 c =
    2.
```

```
-->a^2+b^2==c^2
 годы =
  F
```

Программа отвечает false, потому что значение a^2+b^2 является приближенным

-->abs(a^2+b^2-c^2)<%eps годы = F	Программа отвечает false, потому что запрашиваемая точность является абсолютной
-->abs(a^2+b^2-c^2)/c^2<%eps годы = T	Программа отвечает true, потому что запрашиваемая точность относительна

Для отображения

По умолчанию результаты отображаются с 10 символами, включая десятичную точку и знак. Для отображения большего количества цифр можно использовать функцию **format**. Чтобы получить 20 цифр, введите **format(20)**.

$a = \sqrt{\square}$Давайте вернемся к :

-->a^2 годы = 3.	Здесь 7 знаков после запятой, поэтому точность не видна.
-->format(20) -->a^2 годы = 2.99999999999999956	Здесь 17 знаков после запятой, так что вы можете видеть точность.

Решение дифференциальных уравнений

Здесь мы покажем, как можно найти решения явной системы дифференциальных уравнений. Принцип заключается в сведении к дифференциальным уравнениям порядка 1:

$\{ y'(t) = f(t, y(t)) \; y(t_0) = y_0 t_0 t \, y_0 y(t)$где и представляет собой время, а и - векторы,

затем примените функцию **ode**: y=ode(y0,t0,t,f), где :

- $n_{\mathbf{y0}}$: начальное условие, вектор размерности ,
- **t0**: начальное время,
- **t**: вектор размерности T моментов времени, в которые мы хотим получить решение. Этот вектор должен начинаться с **t0**,
- **f**: функция, определяющая систему в виде :

```
функция yprim=f(t,y)
    yprim(1)=...
    yprim(2)=...
    ....
    yprim(n)=...
endfunction
```

$n \times T$ Решение y - это матрица размерности : $\left(y_1(1)\, y_1(2) \vdots y_1(T)\, y_2(1)\, y_2(2) \vdots y_2(T) \vdots \vdots \vdots y_n(1)\, y_n(2) \ldots y_n(T)\right)$

Пример: Решите дифференциальное уравнение

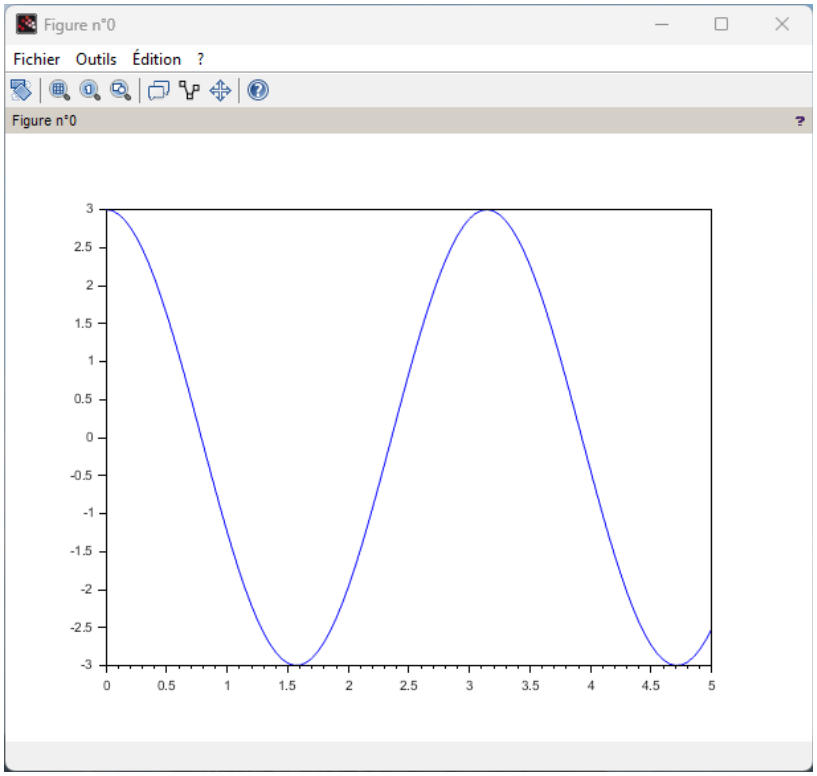
$\{ y$ =-4y y left (0 right) =3, {y} ^ {'} left (0 right) =0

Это уравнение порядка 2 можно свести к системе из двух уравнений порядка 1, поставив :

$Y = \left(Y(1) Y(2)\right) = \left(y\ y'\right), Yprim = \left(Yprim(1) Yprim(2)\right) =$ ¿ и $Yprim(1) = Y(2) Yprim(2) = -4 \times Y(1)$

Комментарий	Редактор Scilab

Мы определяем функцию, которая отображает вектор y' на две переменные t и y (которая является вектором).
 	
Мы задаем значения t для графа
(программа сама выбирает значения t для внутреннего расчета решения).
Мы задаем начальные условия
Мы применяем оду
Постройте интегральную кривую y как функцию t
	`функция yprim=f(t,y)`
	`  yprim(1)=y(2);`
	`  yprim(2)=-4*y(1) ;`
	`endfunction`
	`t0=0; tmax=5;`
	`t=t0:0.05:tmax;`
	`y0=3; yprim0=0;`
	`y=ode([y0;yprim0],t0,t,f);`
	`clf; plot(t,y(1,:))`



Глава 3 - Полезные функции Scilab

Для анализа

- xx**sqrt(x)** возвращает квадратный корень из вещественной положительной части или нуля, а в противном случае - комплексный корень из вещественной положительной части.
- **log(x)** возвращает натуральный логарифм от x, где x - вещественное или комплексное число.
- xx**exp(x)** возвращает экспоненту от вещественного или комплексного числа.
- xx**abs(x)** возвращает абсолютное значение вещественного числа (или модуля, если оно комплексное).
- x**int(x)** возвращает усечение вещественного числа (целое число до десятичной точки).
- $xnn \le x < n+1$**floor(x)** возвращает целую часть вещественного числа (целого числа, такого как).
- $xnn-1 < x \le n$**ceil(x)** возвращает целое число, такое, что .

Для теории вероятностей и статистики

- **factorial(n)** возвращает факториал n, причем n - целое положительное или нулевое число.
- $m \le n$**grand(1,p, "uin",m,n)** возвращает вектор из p целочисленных розыгрышей между m и n, где p - целое положительное число, m и n - целые числа и .
- $a \le b$**grand(1,p, "unf",a,b)** возвращает вектор из p вещественных отрезков между a и b, где p - целое положительное число, a и b - вещественные и .
- n**sum(n)** возвращает сумму значений в векторе (используется для подсчета общего количества людей).
- n**cumsum(n)** возвращает вектор кумулятивно возрастающих значений вектора (используется для вычисления кумулятивно возрастающих чисел).
- v**length(v)** возвращает количество координат вектора .
- v**gsort(v)** возвращает отсортированный вектор чисел или символьных строк в порядке убывания.
- v**gsort(v, "g", "i")** возвращает отсортированный вектор чисел или символьных строк в порядке возрастания.
- v**mean(v)** возвращает среднее значение вектора чисел.
- v**stdev(v)** возвращает стандартное отклонение вектора чисел.
- $vnvn$**bar(v,n,color) строит** гистограмму, где в качестве оси x, оси y выступают линейные векторы одинаковой размерности. nПо умолчанию **bar(n)** строит гистограмму синего цвета с 1,2,3... в качестве абсцисс.
- vv**bar(v,[n1',n2'])** строит двойную гистограмму, где ось x - синяя, ось y - n1, ось y - n2, ось y - зеленая, а линейные векторы n1 и n2 имеют одинаковую размерность.
- $npn \times p$**rand(n,p)** с и положительными целыми числами, возвращает матрицу чисел, взятых случайным образом в диапазоне от 0 до 1.
- **rand()** возвращает действительное число, взятое случайным образом в диапазоне от 0 до 1.
- x**floor(x)** возвращает целую часть вещественного числа . $pp1-p$В частности, если это вещественное число находится в диапазоне от 0 до 1, то **floor(rand()+p)** будет 1 с вероятностью и 0 с вероятностью .

Для отображения и построения графика

- **clf** означает "очистить фигуру" и удаляет фигуру в графическом окне.
- **График** можно использовать для построения кривых или облаков точек в измерении 2.
- $abnnab$**linspace(a,b,n)**, с и вещественными и целыми числами, задает вектор значений, регулярно расположенных между и .
- **scf** можно использовать для открытия или выбора других графических окон.
- **Серф** можно использовать для рисования поверхностей в измерении 3.
- **bar(X,Y)**, где X и Y - векторы, строит гистограмму ряда значений X, а значения Y - их отсчеты.
- **plot(X,Y, "*") строит** облако точек с координатами (X(i),Y(i)) в виде звезд. Вы можете указать цвет.
- **plot(Y, "+") строит** облако точек с координатами (i,Y(i)) в виде креста.
- **disp("Фраза")** выводит то, что написано между кавычками.
- **disp(A)**, где A - матрица чисел, выводит таблицу значений для A.
- x**disp("Фраза "+строка(x))** выводит фразу и значение числа.

Утилиты

- v**unique(v)** возвращает вектор с единственным вхождением его дублированных элементов и сортирует их в порядке возрастания.
- v**sum(v)** возвращает сумму всех элементов вектора или матрицы.
- v**prod(v)** возвращает произведение всех элементов вектора или матрицы.
- v**find(<тест на v>)** возвращает индексы элементов в векторе, которые проверяют тест.
- **disp(x,y,...)** выводит значения своих аргументов в консоль.
- x**string(x)** преобразует число в строку символов.
- nn**format(n)**, где - целое число, большее или равное 2, устанавливает отображение символов, включая знак и запятую.
- $n \times p$**zeros(n,p)** определяет матрицу, содержащую только 0.
- $xymnm \times n(i,j)f(x(i),y(j))$**feval(x,y,f)**, где и - векторы размеров и соответственно, определяет матрицу, элементом которой является .
- **doc(<функция>)** открывает браузер справки на страницу справки по функции.
- **тик** запускает секундомер.

- `toc` останавливает секундомер.