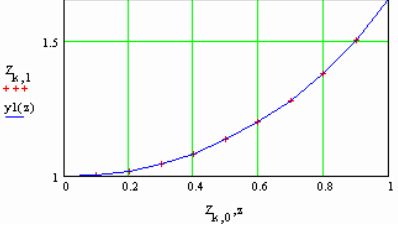


<p>1. Определение численных методов. Классификация численных методов*</p> <p>Численные методы – это алгоритмы вычисления приближенных значений решения в отдельных точках</p> <p>При $q = 0$, $s = 0$, 1 – метод называется одношаговым; $q > 1$ или $s > 1$ – многошаговым; $s = 0$ – явные; $s = 1$ – неявные; $s > 1$ – с забеганием вперед.</p> <p>Т.о., одношаговые методы имеют вид:- явные- неявные.</p>	<p>2. Аппроксимация - это замена исходной функции $f(x)$ на функцию $g(x)$ так, чтобы отклонение $f(x)$ от $g(x)$ в заданной области было наименьшим. Линейная регрессия общего вида реализуется с помощью функции linfit: linfit(VX, VY, F), где VX, VY - координаты исходных точек; F - вектор, содержащий функции $f_i(x)$, записанные в символьном виде. Функция linfit еще называется функцией аппроксимации по методу наименьших квадратов. Вектор VX должен быть возрастающим. Для построения аппроксимирующей зависимости можно воспользоваться встроенной функцией interp (VK, VX, VY, x), Линейная интерполяция При линейной интерполяции Mathcad соединяет существующие точки данных прямыми линиями. Это выполняется функцией linterp, описанной ниже.</p> $linterp(vx, vy, x)$ <p>Эта функция соединяет точки данных отрезками прямых, создавая таким образом ломаную. Функция linterp предназначена для интерполяции, а не экстраполяции.</p> <p>Кубическая сплайн-интерполяция</p> <p>Кубическая сплайн-интерполяция позволяет провести кривую через набор точек таким образом, что первые и вторые производные кривой непрерывны в каждой точке. Эта кривая образуется путем создания ряда кубических полиномов, проходящих через наборы из трёх смежных точек.</p>
<p>3. Метод наименьших квадратов*</p> <p>Наиболее распространенным методом аппроксимации экспериментальных данных является метод наименьших квадратов. Метод позволяет использовать аппроксимирующие функции произвольного вида и относится к группе глобальных методов. Простейшим вариантом метода наименьших квадратов является аппроксимация прямой линией (полиномом первой степени). Этот вариант метода наименьших квадратов носит также название линейной регрессии. Критерием близости в методе наименьших квадратов является требование минимальности суммы квадратов отклонений от аппроксимирующей функции до экспериментальных точек:</p> $\Phi = \sum_{i=1}^n (y_i - f(x_i))^2 \rightarrow \min$ <p>Таким образом, не требуется, чтобы аппроксимирующая функция проходила через все заданные точки, что особенно важно при аппроксимации данных, заведомо содержащих погрешности. Важной особенностью метода является то, что аппроксимирующая функция может быть произвольной. Ее вид определяется особенностями решаемой задачи, например, физическими соображениями, если проводится аппроксимация результатов физического эксперимента</p>	<p>4. Постановка задачи и обзор численных методов решения ОДУ и систем ОДУ*</p> <p>Пусть необходимо найти решение уравнения</p> $y' = f(x, y) \quad (5.1)$ <p>с начальным условием $y(x_0) = y_0$. Такая задача называется задачей Коши.</p> <p>Разложим искомую функцию $y(x)$ в ряд вблизи точки x_0 и ограничимся первыми двумя членами</p> $y(x) = y(x_0) + y'(x)(x - x_0) + \dots$ <p>Разложения $x - x_0 = h$, получаем $y(x) = y(x_0) + f(x_0, y_0) \Delta x$. Эту формулу можно применять многократно, находя значения функции во все новых и новых точках.</p> $y_{i+1} = y_i + f(x_i, y_i)h \quad (5.2)$ <p>Такой метод решения обыкновенных дифференциальных уравнений называется методом Эйлера. Геометрически метод Эйлера означает, что на каждом шаге мы аппроксимируем решение (интегральную кривую) отрезком касательной, проведенной к графику решения в начале интервала. Точность метода невелика и имеет порядок h. Говорят, что метод Эйлера – метод первого порядка, то есть его точность растет линейно с уменьшением шага h. Существуют различные модификации метода Эйлера, позволяющие увеличить его точность. Все они основаны на том, что производную, вычисленную в начале интервала, заменяют на среднее значение производной на данном интервале. Среднее значение производной можно получить (конечно же только приближенно) различными способами. Можно, например, оценить значение производной в середине интервала $[x_i, x_{i+1}]$ и использовать его для аппроксимации решения на всем интервале</p> <p>Оценку значения производной можно улучшить, увеличивая число вспомогательных шагов. На практике наиболее распространенным методом решения обыкновенных дифференциальных уравнений является метод Рунге-Кутты четвертого порядка. Для оценки значения производной в этом методе используется четыре вспомогательных шага. Перечисленные методы можно применять и для решения систем дифференциальных уравнений. Поскольку многие дифференциальные уравнения высших порядков могут быть сведены заменой переменных к системе дифференциальных уравнений первого порядка, рассмотренные методы могут быть использованы и для решения дифференциальных уравнений порядка выше первого.</p>
<p>6. Стандартные функции для решения ОДУ в Mathcad</p> <p>Mathcad имеет ряд встроенных функций для решения дифференциальных уравнений. Для численного решения дифференциального уравнения нужно задать:</p> <ul style="list-style-type: none"> – само дифференциальное уравнение, – начальные условия, – множество точек, в которых нужно найти решение. <p>Наиболее употребляемым численным методом для решения дифференциальных уравнений является метод Рунге-Кутты четвертого порядка. Для реализации этого метода в Mathcad имеется встроенная функция rkfixed, обращение к которой имеет вид</p> $z := rkfixed(y, x1, x2, N, D)$ <p>Аргументы функции имеют следующий смысл:</p> <ul style="list-style-type: none"> y – вектор начальных условий размерности n, где n – порядок дифференциального уравнения; $x1, x2$ – начальная и конечная точки интервала, на котором ищется решение; N – число точек, в которых ищется решение; $D(x, y)$ – функция, которая возвращает значение в виде матрицы-столбца из n элементов. <p>5. Метод Рунге-Кутты для решения ОДУ и систем ОДУ*</p> <p>Решим еще раз задачу Коши для дифференциального уравнения первого</p>	<p>6. Для решения ОДУ порядка $1 > N$ в Mathcad предусмотрены две возможности:</p> <ul style="list-style-type: none"> • вычислительный блок Given/Odesolve (начиная с версии 2000) – в этом случае решение имеет вид функции от t; • встроенные функции решения систем ОДУ, причем уравнения высших порядков необходимо предварительно свести к эквивалентной системе уравнений первого порядка, – в этом случае решение имеет формат вектора. <p>Вычислительный блок для решения ОДУ, реализующий численный метод Рунге-Кутты, состоит из трех частей:</p> <ul style="list-style-type: none"> • Given – ключевое слово; • ОДУ и начальные условия в формате $y(t0) = b$, записанные с помощью логических операторов, которые должны набираться на панели инструментов Boolean (Булевы операторы); • Odesolve(t1) – встроенная функция для решения ОДУ относительно переменной t на интервале $(t0, t1)$ причем $t0 < t1$. <p>7.</p> <p>Дифференциальное уравнение первого порядка — это уравнение, которое не содержит производных выше первого порядка от неизвестной функции</p>

<p>порядка $y' = xy$ методом Рунге-Кутты.</p> <p>Зададим границы изменения x: $x_{\min} := 0$ $x_{\max} := 1$</p> <p>Зададим число точек внутри интервала $n := 10$</p> <p>Зададим начальные условия $y_0 := 1$</p> <p>Определим теперь матрицу производных. Эта матрица тоже состоит только из одного элемента. Этот элемент с точностью до обозначений совпадает с правой частью исходного дифференциального уравнения:</p> $D(x, y) := y_0 \cdot x$ <p>Решаем дифференциальное уравнение $Z := \text{rkfixed}(y, x_{\min}, x_{\max}, n, D)$</p> $k := 0..n \quad y1(x) := \exp\left(\frac{x^2}{2}\right) \quad z := 0, 0.1..1$  <p>Точное аналитическое решение и решение, полученное численно отличаются в точке $x = 1$</p> <p>на $y1(1) - (z^{<1>})_n = 2.636 \cdot 10^{-7}$ Относительная ошибка</p> <p>составляет $\frac{y1(1) - (z^{<1>})_n}{y1(1)} = 1.599 \cdot 10^{-5} \%$</p>	<p>Функция <i>rkfixed</i> на использует для поиска решения метод Рунге-Кутты четвертого порядка. В результате решения получается матрица, имеющая два следующих столбца:</p> <p>Первый столбец содержит точки, в которых ищется решение дифференциального уравнения.</p> <p>Второй столбец содержит значения найденного решения в соответствующих точках.</p> <p>Системы ОДУ первого порядка</p> <p>Для того чтобы решить систему ОДУ первого порядка, необходимо:</p> <p>Определить вектор, содержащий начальные значения для каждой неизвестной функции.</p> <p>Определить функцию, возвращающую значение в виде вектора из n элементов, которые содержат первые производные каждой из неизвестных функций.</p> <p>Выбрать точки, в которых нужно найти приближенное решение.</p> <p>Передать всю эту информацию в функцию <i>rkfixed</i>.</p>
<p>8. Алгоритм решения ОДУ второго порядка в Mathcad. Примеры</p> <p>В качестве примера решим задачу о гармоническом осцилляторе, для которого известно аналитическое решение, и легко может быть оценена точность вычислений. Дифференциальное уравнение второго порядка</p> $y'' + 2\beta y' + \omega^2 y = 0$ <p>преобразуем к системе из двух дифференциальных уравнений первого порядка</p> $y' = x$ $x' = -2\beta x - \omega^2 y$ <p>Пусть декремент затухания $\beta := 0.0$</p> <p>Пусть циклическая частота $\omega := 1$</p> $y := \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ <p>Зададим начальные условия</p> <p>y_0 соответствует начальной координате, а y_1 – начальной скорости. Зададим теперь матрицу D. С учетом того, что искомая величина соответствует нулевому элементу массива y, ее первая производная – первому, а вторая – второму, имеем</p> $D(t, y) := \begin{pmatrix} y_1 \\ -2\beta y_1 - \omega^2 y_0 \end{pmatrix}$ $Z := \text{rkfixed}(y, 0, 5 \cdot \pi, 100, D)$ <p>Представим результаты расчета на графике и сравним их с аналитическим решением</p> $f(x) := y_1 \cdot \exp(-\beta x) \cdot \cos(\omega x)$ $k := 0..100 \quad x := 0, 0.1..5 \cdot \pi$	<p>10. Общая характеристика системы Scilab*, основные возможности</p> <p>Scilab – это кроссплатформенная система компьютерной математики (СКМ), которая предназначена для выполнения научно-технических расчетов, графической интерпретации полученных результатов и визуального моделирования. Эта система имеет удобный пользовательский интерфейс и развитый язык программирования.</p> <p>Сама система Scilab, как и Matlab, предназначена прежде всего для численных расчетов и работы с матрицами. Кроме того, она обладает развитыми средствами программирования, так что ее в какой-то мере можно рассматривать как систему разработки высокотехнологичных приложений.</p> <p>Scilab имеет схожий с MATLAB язык программирования. В состав пакета входит утилита, позволяющая конвертировать документы Matlab в Scilab. Все возможности системы можно классифицировать так:</p> <ul style="list-style-type: none"> - математические; - использования численных методов; - программирование; - графические; - имитационное моделирование; - сервисные. <p>Математические возможности перечислены ниже.</p> <ul style="list-style-type: none"> - Вычисление арифметических и логических выражений. - Вычисление стандартных математических функций. - Операции с векторами и матрицами. - Матричные операции линейной алгебры и т.д. <p>К численным методам относятся:</p> <ul style="list-style-type: none"> - численные методы решения алгебраических уравнений и систем; - методы работы с полиномами; - методы решения обыкновенных дифференциальных уравнений и систем; - методы аппроксимации и интерполяции; - методы минимизации функций и т.д.
<p>11. Сравнительная характеристика возможностей Scilab и Mathcad</p> <p>Все возможности системы можно классифицировать так:</p> <ul style="list-style-type: none"> - математические; - использования численных методов; - программирование; - графические; - имитационное моделирование; - сервисные. <p>Математические возможности перечислены ниже.</p> <ul style="list-style-type: none"> - Вычисление арифметических и логических выражений. - Вычисление стандартных математических функций. 	<p>12. Интерфейс и режимы работы в Scilab*</p> <p>Система имеет несколько режимов работы, каждый из которых поддерживается собственным диалоговым окном (рис.1.1).</p> <ul style="list-style-type: none"> - Командный режим – командное окно. - Программный режим – окно создания и редактирования программных файлов (SCE-файлов). - Графический режим – окно редактирования графиков. - Режим помощи – окно помощи. - Режим демонстрации – окон демонстрационных примеров. <p>При работе в любом из перечисленных режимов могут быть использованы дополнительные информационные окна</p>

<p>- Операции с векторами и матрицами. - Матричные операции линейной алгебры и т.д.</p> <p>Разработчики Mathcad сделали ставку на расширение системы в соответствии с потребностями пользователя. Для этого назначены дополнительные библиотеки и пакеты расширения, которые можно приобрести отдельно и которые имеют дополнительные функции, встраиваемые в систему при установке, а также электронные книги с описанием методов решения специфических задач, с примерами действующих алгоритмов и документов, которые можно использовать непосредственно в собственных расчетах. Кроме того, в случае необходимости и при условии наличия навыков программирования в С, есть возможность создания собственных функций и их прикрепления к ядру системы через механизм DLL.</p> <p>Mathcad изначально создавался для численного решения математических задач, он ориентирован на решение задач именно прикладной, а не теоретической математики, когда нужно получить результат без углубления в математическую суть задачи. Символьное ядро Mathcad, искусственно ограничено (доступно около 300 функций), но этого в большинстве случаев вполне достаточно для решения задач инженерного характера.</p>	<p>Окно рабочей области (Обозреватель переменных) – предназначено для просмотра и редактирования содержимого рабочей области Командный режим Программный режим режим Графический режим</p> <p>памяти, в нем указывается имя переменной (массива или структуры), ее размерность и тип.</p> <p>Окно журнала команд содержит перечень команд, введенных пользователем в командном режиме за текущий и предыдущий сеансы работы с системой.</p> <p>Окно управления файлами (обозреватель файлов) служит для быстрого доступа к файлам при работе с системой.</p> <p>Управлять информационными окнами можно с использованием пункта основного меню «Инструменты».</p>
<p>13. Выполнение базовых вычислений в командном и программном режиме</p> <p>Создание документа в системе Scilab может выполняться в различных режимах, ниже рассматриваются командный и программный режимы. Документ при работе в командном режиме представляет собой последовательность команд пользователя и ответов системы, расположенных в командном окне.</p> <p>Символ --> в окне команд показывает, что система готова к диалогу с пользователем. Командная строка может содержать одну или несколько команд, она завершается нажатием клавиши Enter. Строка реакции системы называется строкой вывода, она показывает результаты выполнения команды либо в стандартной переменной ответа ans, либо в переменной, заданной пользователем, например:</p> <pre>--> 5+3 ans=8 --> b=5+3 b=8</pre> <p>Если необходимо вывести данные на экран дисплея в определенной последовательности, применяется функция disp, которую принято называть оператором вывода. Оператор имеет следующий общий вид: disp(Выражение)</p> <p>Здесь Выражение – это арифметическое, логическое или символьное выражение, частным случаем которого являются константы или переменные любого типа.</p> <p>Каждый новый оператор disp выполняет вывод с новой строки командного окна, например (переменным a, b, k, d уже присвоены числовые значения):</p> <pre>Фрагмент программы c=a-b+k*d; disp ('результат='), disp(c); Командное окно результат= 28</pre>	<p>14. Обработка структурированных данных в Scilab</p> <p>В Scilab можно использовать различные типы структурированных данных, т.е. данных, содержащих несколько элементов. К основным структурированным данным относятся массивы чисел (вектора, матрицы, многомерные массивы).</p> <p>Массив - это последовательность однотипных элементов, снабженных индексами (порядковыми номерами). Вектором принято называть одномерный массив. Матрица - это двумерный массив. Векторы в Scilab делятся на векторы-строки и векторы-столбцы.</p> <p>Занести числа в вектор можно несколькими способами.</p> <p>1) Непосредственный ввод</p> <p>Чтобы задать вектор-строку, значения его элементов следует перечислить в квадратных скобках, разделяя пробелами.</p> <p>Например,</p> <pre>V=[2 3 8 0]</pre> <p>Чтобы задать вектор-столбец, значения его элементов следует перечислить в квадратных скобках, используя для разграничения строк точку с запятой.</p> <pre>V=[2;3;8]</pre> <p>Обращение к элементу вектора выполняется указанием имени вектора и номера элемента в векторе в круглых скобках.</p> <p>Например, V(2) à 3</p> <p>V(3) à 8</p> <p>2) Ввод с использованием диапазона</p> <p>Общий вид диапазона:</p> <pre>xn : dx : xk</pre> <p>xn – начальное значение диапазона; dx – шаг изменения значений диапазона; xk – конечное значение диапазона;</p> <p>Для формирования вектора следует задать:</p> <pre>X=xn:dx:xk</pre> <p>В результате будет сформирован вектор, первый элемент которого равен xn, второй – xn+dx, третий - xn+dx+dx и т.д. Последний элемент будет не больше xk для положительного шага dx, и не меньше xk – для отрицательного. Если величина шага отсутствует, то по умолчанию его значение равно 1 или -1, тогда вид диапазона таков:</p> <pre>X=xn: xk</pre> <p>Система имеет обширный набор стандартных функций и операций по обработке матриц, который позволяет:</p> <ul style="list-style-type: none"> - формировать новые матрицы стандартного вида; - выполнять матричные арифметические операции; - вычислять матричные характеристики и математические функции. <p>Матричные арифметические операции представлены следующими:</p> <p>A+B, A-B матричное сложение и вычитание. Оба операнда этой операции должны иметь одинаковую размерность, если они являются матрицами. Один из операндов может быть скалярной величиной.</p> <p>A*B матричное умножение. Операция выполняется по правилам матричного умножения, число столбцов матрицы A должно быть равно числу строк матрицы B.</p> <p>Операция «апостроф» ' вычисляет транспонированную матрицу.</p> <p>Система содержит стандартные функции, позволяющие вычислять различные характеристики матриц:</p> <pre>det(A) вычисляет определитель матрицы; trace(A) вычисление следа матрицы; rank(A) вычисление ранга матрицы; inv(A) вычисление обратной матрицы.</pre> <p>Над массивами можно выполнять различные операции, заданные системными функциями.</p> <pre>max(A) - вычисление максимального элемента массива; min(A) - вычисление минимального элемента массива; sum(A) - вычисление суммы элементов массива; prod(A) - вычисление произведения элементов массива; mean(A) - вычисление среднего значения элементов массива. [Amax, Nmax]= max(A) - вычисление максимального элемента в массиве и его номера.</pre>

15. Сравнительная характеристика обработки структурированных данных в Scilab и Mathcad.

В Scilab можно использовать различные типы структурированных данных, т. е. данных, содержащих несколько элементов. К основным структурированным данным относятся массивы чисел (вектора, матрицы, многомерные массивы). Векторы в Scilab делятся на векторы-строки и векторы-столбцы. Занести числа в вектор можно несколькими способами:

1) Непосредственный ввод. Чтобы задать вектор-строку, значения его элементов следует перечислить в квадратных скобках, разделяя пробелами или запятыми. Например, $V = [2 \ 3 \ 8 \ 0]$. Чтобы задать вектор-столбец, значения его элементов следует перечислить в квадратных скобках, используя для разграничения строк точку с запятой. $V = [2;3;8]$. Обращение к элементу вектора выполняется указанием имени вектора и номера элемента в векторе в круглых скобках. 2) Ввод с использованием диапазона. Общий вид диапазона: $xn : dx : xk$. xn – нач значение диапазона; dx – шаг изменения значений диапазона; xk – кон значение диапазона. Для формирования вектора следует задать: $X = xn:dx:xk$ В результате будет сформирован вектор, первый элемент которого равен xn , второй – $xn+dx$, третий – $xn+dx+dx$ и т. д. Последний элемент будет не больше xk для положительного шага dx , и не меньше xk – для отрицательного. Если величина шага отсутствует, то по умолчанию его значение равно 1 или -1, тогда вид диапазона таков: $X = xn : xk$.

Чтобы задать матрицу, значения ее элементов следует перечислить в квадратных скобках, разделяя элементы в строках пробелами или запятыми, а для разграничения строк использовать точку с запятой. Для указания отдельного элемента матрицы после имени матрицы в круглых скобках указываются два индекса: номер строки и номер столбца. Для формирования новых матриц стандартного вида применяются следующие системные функции: $\text{rand}(M,N)$, $\text{ones}(M,N)$, $\text{zeros}(M,N)$, $\text{diag}(V)$. Матричные арифметические операции представлены следующими: $A+B$, $A \cdot B$ – матричное сложение и вычитание. Один из операндов может быть скалярной величиной. A^*B – матричное умножение. X^P – возведение матрицы в степень. Эта операция является ошибочной, если оба операнда – матрицы. В Scilab существуют матричные операции, которые выполняются над каждым элементом матрицы, это такие операции, как: $.*$ – поэлементное матричное умножение; $./$ – поэлементное левое деление матриц; $./$ – поэлементное правое деление матриц; $.^$ – поэлементное возведение матрицы в степень. Оба операнда этих операций должны иметь одинаковую размерность, или один из них должен являться скалярной величиной. Операция «апостроф» $'$ вычисляет транспонированную матрицу. Система содержит стандартные функции, позволяющие вычислять различные характеристики матриц: $\text{det}(A)$ – вычисление определителя матрицы; $\text{trace}(A)$ – вычисление следа матрицы; $\text{rank}(A)$ – вычисление ранга матрицы; $\text{inv}(A)$ – вычисление обратной матрицы. Над массивами можно выполнять различные операции, заданные системными функциями. $\text{max}(A)$ – вычисление максимального элемента массива; $\text{min}(A)$ – вычисление минимального элемента массива; $\text{sum}(A)$ – вычисление суммы элементов массива; $\text{prod}(A)$ – вычисление произведения элементов массива; $\text{mean}(A)$ – вычисление среднего значения элементов массива. $[\text{Amax}, \text{Nmax}] = \text{max}(A)$ – вычисление максимального элемента в массиве и его номера.

Обработка структурированных данных в MathCad.

Дискретной называется переменная, содержащая несколько значений, изменяющихся от начального до конечного на величину постоянного шага. Дискретная переменная может быть задана двумя способами: 1) $a := a1, a2 .. an$ 2) $a := a1 .. an$ где a – имя дискретной переменной, $a1$ – ее начальное значение, $a2$ – ее второе значение, an – ее конечное значение. В системе MathCAD в основном используются массивы двух типов: одномерные (векторы) и двумерные (матрицы). Каждый элемент вектора или матрицы имеет порядковый номер в массиве. Отсчет номеров начинается с того значения, которое содержится в системной переменной **ORIGIN**. По умолчанию эта переменная имеет значение 0, для изменения значения нужно задать. Векторы и матрицы можно задавать различными способами: с помощью кнопки с изображением матриц на наборной панели математических инструментов; как переменную с индексами перечислением элементов массива с разделением запятой; с помощью аналитического выражения. Массивы могут использоваться в выражениях целиком или поэлементно. Для обращения к элементам массивов нужно указать числовые значения индексов элементов в подстроичнике после имени массива. При выполнении расчетов можно обращаться к конкретной строке или столбцу матрицы с помощью верхнего индекса или нижних индексов. Существует ряд операций над матрицами и векторами, а также встроенных векторных и матричных функций. Введем следующие обозначения: V – вектор, M – матрица.

17. Построение графиков в Scilab, графики кусочно-непрерывных функций, примеры.

Графические объекты в Scilab строятся в специальном графическом окне (figure). Одновременно может быть открыто несколько таких окон, каждому из которых присваивается номер. Для перехода к имеющемуся окну с номером N или открытия нового графического окна необходимо ввести команду figure(N). Кроме того, первое обращение к графической команде автоматически вызывает появление графического окна, которому присваивается номер 0. Для построения графиков функций одной переменной в декартовой системе координат используются различные формы команды plot, которая рисует графики функций по ряду точек, соединяя их отрезками прямых. Команда plot(X,Y) – строит график функции, координаты точек которой берутся из векторов одинаковой размерности X и Y. Если Y – матрица, то строится семейство графиков по данным, содержащимся в столбцах матрицы. Команда plot(Y) – строит график зависимости, значения ординат которой

16. Обзор основных операторов условия и цикла в Scilab

Для программирования разветвляющихся алгоритмов в Scilab существуют оператор условия и оператор выбора. Условный оператор представлен в нескольких формах и имеет следующий общий вид. **Полная форма 1:** if LVB then ОПЕРАТОР1 else ОПЕРАТОР2, end. **Полная форма 2:** if LVB then ОПЕРАТОР1, else if LVB2 then ОПЕРАТОР2, else ОПЕРАТОР, end. **Краткая форма:** if LVB then ОПЕРАТОР1 end. LVB, LVB1, LVB2, LVB3 – логические выражения; ОПЕРАТОР1, ОПЕРАТОР2, ОПЕРАТОР – любые операторы или группы операторов языка Scilab. Разделителями в операторах могут быть запятая или точка с запятой. Если ОПЕРАТОР расположен в следующей строке, то разделитель перед ним ставить не обязательно. Порядок выполнения оператора условия следующий. Полная форма 1 • Вычисляется значение логического выражения. • Если логическое выражение истинно, то выполняется оператор, стоящий после слова then, а затем следующий за оператором if оператор. • Если логическое выражение ложно, то выполняется оператор, стоящий после слова else, а затем следующий за оператором if оператор. Краткая форма • Вычисляется значение логического выражения. • Если логическое выражение истинно, то выполняется оператор, стоящий после слова then, а затем следующий за оператором if оператор. • Если логическое выражение ложно, то выполняется следующий за оператором if оператор.

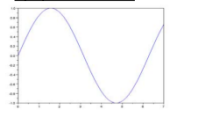
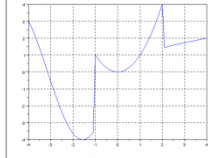
Операторы цикла предназначены для программирования циклических алгоритмов, они изменяют естественный ход выполнения программы и относятся к операторам управления. Операторы цикла в Scilab можно классифицировать следующим образом: – оператор цикла с параметрами for; – оператор цикла с предусловием while. Оператор цикла for предназначен для программирования циклических алгоритмов, когда переменная цикла явно выражена и изменяется от начального значения до конечного значения с постоянным шагом. Общий вид оператора цикла for: for $x = xn : dx : xk$ ОПЕРАТОР, end. Здесь: x – переменная цикла; xn – начальное значение переменной цикла; dx – шаг изменения переменной цикла; xk – конечное значение переменной цикла; ОПЕРАТОР – любой оператор или группа операторов рабочей части цикла. Порядок выполнения оператора цикла: – проверяется условие $xn \leq xk$; – если условие не выполняется, то оператор цикла прекращает свою работу, и выполняется следующий за ним оператор программы; – если условие выполняется, то переменной цикла присваивается ее начальное значение $x=xn$; – выполняется оператор рабочей части цикла; – до тех пор, пока $x \leq xk$, переменная цикла увеличивается на шаг dx и выполняется рабочая часть цикла. Количество повторений цикла вычисляется по формуле $|xk - xn|/dx + 1$.

Оператор while предназначен для программирования любых циклов, где проверка условия повторения цикла выполняется перед выполнением рабочей части цикла. Общий вид: while LVB, ОПЕРАТОР, end Здесь: LVB – логическое выражение; ОПЕРАТОР – любой оператор или группа операторов рабочей части цикла. Порядок выполнения оператора следующий: – проверяется истинность логического выражения; – до тех пор, пока оно истинно, выполняется оператор рабочей части цикла; – если логическое выражение стало ложным, то выполняется следующий за оператором цикла оператор программы.

18. Форматирование графиков, многооконный вывод графиков в Scilab.

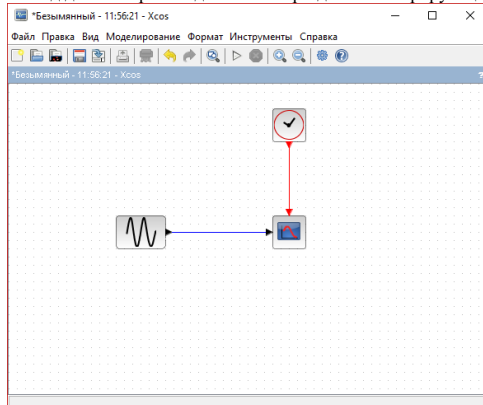
Команда plot(X,Y,S) аналогична команде plot(X,Y), в которой формат линии графика можно задавать с помощью строковой кон- станты S. Символы, которые могут использоваться в параметре S, приведены в табл.

Тип линии	Тип точки	Цвет линии
- Сплошная	.	Точка
: Двойной пунктир	o	Окружность
-. Штрих-пунктир	x	Крест
-- Штриховая	+	Плюс
	*	Звездочка
	s	Квадрат
	d	Ромб
	v	Треугольник
	u	Желтый
	m	Фиолетовый
	c	Голубой
	r	Красный
	g	Зеленый
	b	Синий
	w	Белый
	k	Черный

<p>берутся из вектора Y, а значения абсцисс представляют собой индексы соответствующих элементов вектора. Для построения графиков двух функций – $\sin(x)$ и $\cos(x)$, значения функции которых содержатся в векторах y1 и y2, а значения аргумента x хранятся в векторе x,</p> <p>Например, для построения графика функции $y=\sin(t)$ нужно задать следующий фрагмент программы.</p> <div style="display: flex; align-items: center;"> <div style="border: 1px solid black; padding: 5px; margin-right: 10px;"> <p>Фрагмент программы</p> <pre>t=0:0.01:7; y=sin(t); plot(t,y) или t=0:0.01:7; plot(t,sin(t))</pre> </div> <div style="border: 1px solid black; padding: 5px;"> <p>Графическое окно</p>  </div> </div> <p>Построение графиков кусочно-непрерывных функций.</p> <div style="display: flex; align-items: center;"> <div style="border: 1px solid black; padding: 5px; margin-right: 10px;"> <p>Программа</p> <pre>figure(1) X=4:0.1:4, i=1 while i<=length(X) if X(i)>2 then y=sqrt(X(i)), elseif (X(i)>=-1) & (X(i)<=2) then y=X(i)^2 else y=4*sin(X(i)) end Y(i)=y i=i+1 end plot(X,Y), xgrid()</pre> </div> <div style="border: 1px solid black; padding: 5px;"> <p>Результат выполнения</p>  </div> </div> <p style="text-align: right; font-size: small;">Активация Win</p>	<p>Команды xgrid () позволяют задавать построение сетки на поле графика. Заголовок графика и надписи осей графика можно вывести с помощью команды xtitle(title,xstr,ystr), где title – символьная константа, содержащая название графика; xstr – символьная константа, содержащая название оси X; ystr – символьная константа, содержащая название оси Y. Идентификацию кривых графика (создание легенды) можно выполнить с использованием команды legend, например, legend("График Y(X)","График Y1(X1)")</p> <p>Для создания в графическом окне нескольких графических областей для вывода графиков применяется команда subplot(m, n, p), которая разбивает графическое окно на m×n областей, где m – число областей по вертикали, n – число областей по горизонтали, p – номер области, в которую будет выводиться текущий график (области отсчитываются последовательно по строкам).</p>
<p>19. Формирование оператора-функции и подпрограммы-функции в Scilab, примеры.</p> <p>В общем виде обращение к функции имеет вид:</p> <p>имя_переменной=имя_функции(арг1 [,arg2,...])</p> <p>Здесь имя_переменной – переменная, куда запишутся результаты работы функции; имя_функции – имя встроенной или введенной ранее пользователем функции; арг1, арг2 и т.д. – аргументы функции.</p> <p>Их рассмотрение начнем с математических функций, как наиболее распространенных в инженерной практике. Одним из основных операторов в Scilab является оператор присваивания. В программе этот оператор выполняет следующие функции: присваивает переменной, стоящей слева от знака «=» значение выражения, стоящего справа. Общий вид оператора присваивания: Имя_переменной = Выражение. Пример: A = cos(x)+c-d^2*p^2+4.92</p> <p>В Scilab в качестве оператора ввода используется функция input, которую, в силу ее значимости при программировании, принято называть оператором. Она имеет следующий общий вид: ИМЯ = input(Символьная константа). Пример: S=input("Задайте площадь").</p> <p>Если необходимо вывести данные на экран дисплея в определенной последовательности, применяется функция disp, которую принято называть оператором вывода. Оператор имеет следующий общий вид: disp(Выражение) Здесь Выражение – это арифметическое, логическое или символьное выражение, частным случаем которого являются константы или переменные любого типа. Каждый новый оператор disp выполняет вывод с новой строки командного окна, например: disp ("результат="), disp(c);</p>	<p>20. Линейная и сплайновая интерполяция данных в Scilab.</p> <p>Интерполяция использует значения некоторой функции, заданные в ряде точек, чтобы предсказать значения функции между ними. Применение линейной интерполяции в Scilab</p> <p>С помощью команды interpIn</p> <p>Синтаксис [y]=interpIn(xyd,x)</p> <p>Параметры</p> <p>xyd : двухстрочная матрица (ху координаты точек)</p> <p>х : вектор (абсциссы)</p> <p>у : вектор (значения по оси у)</p> <p>Команда interpIn интерполирует функцию в интервалах между дискретными узлами, заданных с помощью матрицы xyd с помощью отрезков прямой и возвращает значение интерполирующей ломаной в дискретных точках, заданных вектором х.</p> <p>Построение кубического сплайна в Scilab состоит из двух этапов: вначале вычисляются коэффициенты сплайна с помощью функции d=splin(x,y), а затем рассчитывается значения интерполяционного полинома в точке y=interp(t,x,y,d).</p> <p>Функция d=splin(x,y) имеет следующие параметры:</p> <p>х – строго возрастающий вектор, состоящий минимум из двух компонент;</p> <p>у – вектор того же формата, что и х;</p> <p>d – результат работы функции, коэффициенты кубического сплайна</p> <p>Для функции y=interp(t,x,y,k) параметры х, у и d имеют те же значения, параметр t это вектор абсциссы, а у - вектор ординат, являющихся значениями кубического сплайна в точках х.</p>
<p>21. Аппроксимация данных в Scilab по методу наименьших квадратов.</p> <p>Метод наименьших квадратов позволяет по экспериментальным данным подобрать такую аналитическую функцию, которая проходит настолько близко к экспериментальным точкам, насколько это возможно. Идея метода наименьших квадратов заключается в том, что функцию $Y = f(x, a_0, a_1, \dots, a_k)$ необходимо подобрать таким образом, чтобы сумма квадратов отклонений измеренных значений y_i от расчетных Y_i была наименьшей:</p> $S = \sum_{i=1}^n (y_i - f(x_i, a_0, a_1, \dots, a_k))^2 \min$ <p>Задача сводится к определению коэффициентов. Для реализации этой задачи в Scilab предусмотрена функция</p> <p>[a,S]=datafit(F,z,c),</p> <p>где F функция, параметры которой необходимо подобрать; z матрица исходных данных; c вектор начальных приближений; a вектор коэффициентов; S сумма квадратов отклонений измеренных значений от расчетных. Команда datafit оптимально описывает экспериментальные данные полиномами не ниже второй степени.</p>	<p>22</p> <p>В Scilab используется динамический способ создания интерфейсных компонентов. Он заключается в том, что на стадии выполнения программы могут создаваться (и удаляться) те или иные элементы управления (кнопки, метки, флажки и т. д.) и их свойствам присваиваются соответствующие значения. Для создания любого интерфейсного компонента с заданными свойствами используется функция uicontrol, возвращающая указатель на формируемый компонент:</p> <p>C=uicontrol(F, 'Style', 'тип_компонента', 'Свойство_1', Значение_1, 'Свойство_2', Значение_2,..., 'Свойство_k', Значение_k);</p> <p>Здесь C — указатель на создаваемый компонент;</p> <p>F — указатель на объект, внутри которого будет создаваться компонент (чаще всего этим компонентом будет окно); первый аргумент функции uicontrol не является обязательным, и если он отсутствует, то родителем (владельцем) создаваемого компонента является текущий графический объект — текущее графическое окно;</p> <p>'Style' — служебная строка Style, указывает на стиль создаваемого компонента (символьное имя);</p> <p>'тип_компонента' — определяет, к какому классу принадлежит создаваемый компонент, это может быть PushButton, Radiobutton, Edit, StaticText, Slider, Panel, Button Group, Listbox или другие компоненты, это свойство будет указываться для каждого из компонентов;</p> <p>'Свойство_k', Значение_k — определяют свойства и значения отдельных компонентов, они будут описаны ниже конкретно для каждого компонента. У существующего интерфейсного объекта можно изменить те или иные свойства с помощью функции set:</p> <p>set(C,'Свойство_1',Значение_1, 'Свойство_2', Значение_2, ..., 'Свойство_k', Значение_k)</p> <p>Здесь C — указатель на динамический компонент, параметры которого будут меняться. C может быть и вектором динамических элементов, в этом случае функция set будет задавать значения свойств для всех объектов C(i); 'Свойство_k', Значение_k — изменяемые параметры и их значения. Получить значение параметра компонентов можно с помощью функции get следующей структуры:</p>

	<p>get(C,'Свойство')</p> <p>25</p> <p>Источники сигналов и воздействий</p> <p>Раздел предназначен для входных сигналов обеспечивающих работу модели</p> <p>Имеют только один выход и не имеют входов</p> <p>CONST_m – константа</p> <p>GENSIN_f - генератор синусоидального сигнала</p> <p>STEP_FUNCTION - генератор ступенчатого сигнала</p> <p>RAND_m – формирует случайный сигнал</p> <p>Регистрирующие устройства</p> <p>Раздел позволяет визуализировать получаемые при моделировании результаты и проконтролировать правильность работы отдельных блоков и системы в целом</p> <p>Имеют только входы и не имеют выходов</p> <p>CSCOPE осциллограф</p> <p>AFFICH_m цифровой дисплей</p> <p>CSCOPYXY построение графика одного сигнала к функции другого (имеет два входа, верхний – сигнал по оси x, нижний сигнал – по оси y)</p> <p>Система с непрерывным временем</p> <p>Раздел предназначен для создания линейных стационарных звеньев, описания дифференциальных уравнений</p> <p>DERIV моделирование дифференциального звена</p> <p>INTEGRAL_f моделирование интегрирующего звена</p> <p>Маршрутизация сигналов</p> <p>Раздел предназначен для пересылки, переключения, объединения и разделения сигналов.</p> <p>DEMUX разделяет входной векторный сигнал на отдельные составляющие</p> <p>Математические операции</p> <p>BIGSOM_f суммирование сигналов</p> <p>PRODUCT умножение (деление) входных сигналов</p> <p>TrigFun преобразование входного сигнала с помощью тригонометрических функций</p> <p>Порты и подсистемы</p> <p>Раздел предназначенный для разработки сложных моделей содержащих модели более низкого уровня</p> <p>SUPER_f силовая система может быть представлена как совокупность блоков, внутренняя структура которых скрыта</p>
<p>23. Командная кнопка типа PushButton создается с помощью функции uicontrol, в которой параметру 'Style' необходимо присвоить значение 'pushbutton'. По умолчанию она не снабжается никакой надписью, имеет серый цвет и располагается в левом нижнем углу фигуры, Надпись на кнопке можно установить с помощью свойства String, расположение кнопки задаётся функцией Position</p> <p>Button=icontrol('style','pushbutton','string','Кнопка','position',[50,50,100,20]);</p> <p>Щелчок по кнопке генерирует событие CallBack, которое указывается как параметр функции uicontrol. Значением параметра CallBack является строка с именем функции, вызываемой при щелчке по кнопке. В этом случае функция uicontrol становится такой</p> <p>Button=icontrol('style','pushbutton','string','Button','CallBack','Function')</p> <p>Следующим наиболее часто используемым компонентом является метка — текстовое поле для отображения символьной информации. Для определения метки значения параметра 'Style' в функции uicontrol должно иметь значение 'text'. Компонент предназначен для вывода символьной строки (или нескольких строк). Выводимый на метку текст— значение параметра 'String' — может быть изменен только из программы.</p> <p>uicontrol('Style','text','Position',[10,130,150,20],'String','Метка')</p> <p>Одним из основных свойств метки является горизонтальное выравнивание текста, которое определяется свойством HorizontalAlignment. Это свойство может принимать одно из следующих значений:</p> <p>left — выравнивание текста по левому краю;</p> <p>center — выравнивание текста по центру (значение по умолчанию);</p> <p>right — выравнивание по правому краю.</p> <p>hSt1=icontrol('Style','text','Position',[30,30,150,20],... 'String','Метка 1');</p> <p>set(hSt1,'BackgroundColor',[1 1 1]);</p> <p>set(hSt1,'HorizontalAlignment','left')</p> <p>Рассмотрим еще два компонента — переключатель и флажок, которые позволяют переключаться между состояниями или выключать одно из свойств. У флажка свойство 'Style' принимает значение 'checkbox', у переключателя свойство 'Style' должно быть установлено в 'radiobutton'. Индикатором альтернативных комбинаций является переключатель (Radiobutton), который также создается с помощью функции uicontrol.</p> <p>hFig=figure();</p> <p>R=icontrol('Style','radiobutton','String','name','value',1,'Position',[25,150,70,30]);</p> <p>При создании переключателя должно быть задано его состояние (параметр 'value'), переключатель может быть активен (значение 'value' равно 1) или нет (значение 'value' равно 0). Задать значение свойства 'value' можно также и с помощью функции set. Например:</p> <p>set(Rb,'value',0)</p>	<p>24</p> <p>Создание моделей в пакете Xcos основывается на использовании технологии Drag-and-Drop (перетяни и оставь). В качестве «кирпичиков» при построении модели используются визуальные блоки (модули), которые хранятся в библиотеке Xcos. Xcos-модель может иметь иерархическую структуру, т. е. состоять из моделей более низкого уровня, причем количество уровней иерархии практически не ограничено. На протяжении моделирования есть возможность наблюдать за процессами, которые происходят в системе. Для этого используются специальные блоки («обзорные окна»), входящие в состав библиотек Xcos. Состав библиотек Xcos может быть пополнен пользователем за счет разработки собственных блоков.</p> <p>Последовательность создания и обработки модели</p> <ol style="list-style-type: none"> 1 Создание файла модели 2 Выбор блоков модели 3 Настройка параметров блока 4 Соединение блоков модели 5 Запуск модели на выполнение <p>Концепции моделирования в пакете Xcos</p> <p>Последовательность создания и обработки модели</p> <ul style="list-style-type: none"> • Схема технического устройства или процесса представляется в виде блочной диаграммы • Каждый блок диаграммы должен содержаться в библиотеке блоков пакета • Блоки соединяются между собой линиями потока • Блоки настраиваются на нужные параметры • Каждый блок автоматически формирует программу на внутреннем языке системы Scilab, которая запускается на выполнение при запуске модели

26. Ввод данных происходит в блоки раздела Источники сигнала
Вывод данных происходит в блоки раздела Регистрирующие устройства



27

Последовательность создания и обработки модели

- Схема технического устройства или процесса представляется в виде блочной диаграммы
- Каждый блок диаграммы должен содержаться в библиотеке блоков пакета
Блоки могут иметь различное число входов и выходов называемых портами
Порты делятся на обычные (для ввода/вывода данных, чёрного цвета) и управляющие (для ввода/вывода управляющей информации, красного цвета)
- Блоки соединяются между собой линиями потока
- Блоки настраиваются на нужные параметры
- Каждый блок автоматически формирует программу на внутреннем языке системы Scilab, которая запускается на выполнение при запуске модели

Перед выполнением расчёта необходимо задать параметры расчёта в пункте меню Моделирование - Установка

28

$$\begin{cases} \frac{di}{dt} = \frac{e - R * i - U_c}{L} \\ \frac{dU_c}{dt} = \frac{i}{C} \end{cases}$$

