## Тема 3.3. Символьные (аналитические) вычисления и алгебраические преобразования

**3.3.1. Введение в символьные вычисления**

**3.3.2. Создание и работа с символьными переменными, выражениями и**

**функциями**

**3.3.3. Упрощение выражений, подстановки и разложение на множетели**

**3.3.4. Вычисление сумм рядов и произведений**

**3.3.5. Вычисление пределов**

**3.3.6. Вычисление производной**

**3.3.7. Графические возможности Simbolic**

### 3.3.1. Введение в символьные вычисления

Система является **MatLab** самой крупной системой компьютерной математики, ориентированной на матричные и численные вычисления. Однако MatLab имеет также и средства аналитических вычислений. Пакет **Symbolic Math Toolbox** добавил системе MatLab качественно новые возможности, связанные с выполнением символьных вычислений и преобразований, которые были доступны только в системе принципиально иного класса, относящихся к компьютерной алгебре. Теперь MatLab, с учетом новых средств, становится в полной мере *универсальной* системой. Последняя реализация системы символьной математики Maple в своем ядре и в расширениях имеет около 3000 функций. Система MatLab с пакетом **Symbolic**, включающим в себя чуть больше сот­ни символьных команд и функций, намного уступает Maple по ко­личеству таких команд и функций. Однако в данный пакет включе­ны лишь наиболее важные и широко распространенные функции. Кроме того, есть специальная команда, которая дает доступ к ядру Maple**,** что заметно расширяет круг используемых функций.

С помощью команды **help symbolic** можно получить перечень входящих в пакет команд и функций. Для получения справки по любой команде или функции можно использовать ко­манду **help sym / name.m**, где **name** – это имя соответствующей команды или функции, а **name.m** — имя **m-файла**, задающего данную команду или функцию. Пакет **Symbolic Math Toolbox** включает следующие типы математических вычислений, которые приведены в таблице 3.3.1.

Таблица 3.3.1.

|  |  |
| --- | --- |
|  |  |
| Вычисления | Дифференцирование, интегрирование, пределы, суммы и произведения, разложение в ряд Тейлора |
| Линейная алгебра |  |
|  |  |
|  |  |
|  |  |
|  |  |

С демонстрационными примерами пакета **Symbolic** можно ознакомиться с помощью директории **Symbolic Toolbox**.

### 3.3.2. Создание и работа с символьными переменными, выражениями и функциями

Поскольку переменные системы MatLab по умолчанию не определены и традиционно задаются как векторные, матричные, числовые и т. д., то есть не имеющие отношения к символьной математике, для реализации символьных вычислений нужно прежде всего, позаботиться о создании специальных **символьных переменных***.* В про­стейшем случае их можно определить как строковые переменные, заключив имена в апострофы. Например,

|  |
| --- |
|  |
| >> sin(x)^2 + соs(х)^2  ??? Undefined function or variable 'x'.  >> sin('x')^2 + cos('x')^2  ans =  1  >> |

Мы видим, что система MatLaв «возмутилась» нашей не­брежностью и сообщила, что функция или переменная **х** не опред­елена и ни о каких вычислениях синуса и косинуса речи быть не может. Вместе с тем она подсказала, как надо поступить - заключить имя переменной в апострофы, ибо таким образом система получает информацию о необходимости включить символь­ный режим вычислений. Поэтому во второй раз получен вполне осмысленный результат - сумма квадратов синуса и косинуса пе­ременной **'х'** выдана равной **1**.

Таким образом для работы с командами ядра Mаple в MatLab необходимо создать новый символьного объекта, который фактически является строковой переменной. То есть для проведения аналитических (символьных) операций нужно, чтобы соответствующие переменные были предварительно объявлены с помощью функции **sym():**

**имя\_переменной = sym(**'**имя\_переменной**'**)**

Например,

|  |
| --- |
|  |
| >>х = sym ('x')  x=  x  >>a=sym ('alfa')  a  alfa  >> |

Рассмотрим пример иллюстрирующий различие между стандартными типами данных MatLabсимвольных объектов.

|  |
| --- |
|  |
| >>sqrt(2)  ans=  1.4142  >>a=sqrt(sym(2))  a=  2^(1/2)  >>double(a)  ans=  1.4142  >> |

Пакет Symbolic может работать с числами различных форматов, применяемых в MatLab**.** Рассмотрим следующий пример:

|  |
| --- |
|  |
| >>1/2+3/4  ans=  1.2500  >>sym(1/2+3/4)  ans=  5/4  >>pi/2  ans=  1.5708  sym(pi/2)  ans=  pi/2  >>sin(pi/3)  ans=  0.8660  >>sym(sin(pi/3))  ans=  sqrt(3/4)  >>exp(1)  ans=  2.7183  >>sym(exp(1))  ans=  6121026514868074\*2^(-51)  >> |

Необходимо обратить внимание на то, что MatLab производить вычисления с   
представлением чисел в формате с плавающей точкой с двойной точности (**double** ), в то время как пакет **Symbolic** стремиться представить числа в наиболее точном виде, то есть в целом или рациональном виде.

Необходимо также обратить внимание и на то, что результат символьных   
преобразований отображается при выводе без отступа, которым сопровождается выдача иных результатов. Это позволяет сразу опознать его как символьный, в отличии от обычных численных результатов.

Для создания группы символьных объектов служит команда **syms:**

**syms argl arg2 ...**

Например, набор команд

|  |
| --- |
|  |
| >>a=sym('a ');  >>b=sym('b');  >>c=sym('c');  >>x=sym('x ');  >> |

соответствует следующей команде

|  |
| --- |
|  |
| >>syms a b c x  >> |

Рассмотрим еще один пример на применение команды **syms.**

|  |
| --- |
|  |
| >>syms x y1 y2  >>y1=sin(x)^2;  >>y2=cos(x)^2;  >>y1+y2  ans=  sin(x)^2+cos(x)^2  >> |

Для работы с действительными и комплексными числами существует ряд возможностей. Прежде всего это применение опций **'real'**,  **'unreal'**, **'imag'**. Например,

|  |
| --- |
|  |
| >>x=sym('x' , real);  >>y=sym('y', real);  >>z=x+i\*y  z=  x+i\*y  >>real(z)  ans=  x  >>imag(z)  ans=  y |

Функции **real(z)** и **imag(z)** обеспечивают выделение действительной и мнимой частей комплексного числа **z**.

Снятие наложенных ограничений на возможные значения переменных, например, определенной выше действительной переменной **х**, осуществляется командой

|  |
| --- |
|  |
| >>sym 'x', 'unreal'  % или  >>syms x unreal |

что делает переменную х не вещественной.

Из вышеприведенных примеров видно, что создание символьного выражения осуществляется командой

**sym('символьное\_выражение ')**

Символьные выражения стоятся на основе переменных, констант, арифметических операторов и функций – как встроенных, так и определяемых пользователем. В выражениях могут использоваться арифметические операции. В общем случае это векторные и матричные операции. Например,

|  |
| --- |
|  |
| >> sym x a b  >> f=(sin(x)+a)^2\*(cos(x)+b^2/sqrt(abs(a+b))  f=(sin(x)+a)^2\*(cos(x)+b^2/sqrt(abs(a+b))  >> |

В математических выражениях могут использоваться как обычные, так и символьные переменные. Функция **findsym(S)** позволяет выделить символьные переменные в составе выражения **S**. Она возвращает в алфавитном порядке список всех символьных переменных выражения **S**. При отсутствии таковых возвращается пустая строка. Например:

|  |
| --- |
|  |
| >>sym x, z, y  >> а =2; b = 4 ;  >> findsym (a\*x^2 + b\*y + z)  ans =  х, у, z  >>findsym(a + b + x + y + z,)  ans =  х, у, z  >> |

Символьные переменные могут являться элементами матриц и векторов, причем с ними можно выполнять различные операции. Например,

|  |
| --- |
|  |
| >>syms a b c d  >>M=[a b; c d]  M=  [ a, b]  [ c, d]  >> inv(M)  ans=  [ d/(a\*d-b\*c), -b/(a\*d-b\*c)]  -c/(a\*d-b\*c), a/(a\*d-b\*c)]  >> |

MatLab в отличие от современных систем MathCad, Maple или Mathematica, пока не способна выводить выражения и результаты их преобразований в естественной математической форме с использованием общеприня­тых спецзнаков для отображения интегралов, сумм, произведений и т. д. Тем не менее некоторые ограниченные текстовым форматом воз­можности близкого к математическому виду вывода обеспечивает функция **pretty(S)**. Она позволяет вывести выражения **S** в формате, приближенном к математическому формату, например,

|  |
| --- |
|  |
| >> syms x a b  >> f=(sin(x)+a)^2\*(cos(x)+b)^2/sqrt(abs(a+b))  f=(sin(x)+a)^2\*(cos(x)+b)^2/sqrt(abs(a+b))  >> pretty(f)  2 2  (sin(x) + a) (cos(x) + b)  -------------------------------  1/2  | a + b |  **>>** |

Символьные операции позволяют находить или точные значения выражения или значения со сколь угодно большой точностью. Как известно для преобразования значения числовой переменной в символьную служит функция **sym().** Причем при переходе от числовых к символьным выражениям по умолчанию используется запись чисел в виде рациональной дроби. Использование рациональных дробей при выполнении символьных вычислений означает, что всегда получается точный результат, не содержащий погрешность округления.

Для вычисления символьного выражения с заданной точностью предназначена функция **vpa()**:

**vpa(символьная \_переменная, число значащих цифр)**

По умолчанию результат этой функции содержит 32 значащие цифры.

Второй аргумент **vpa()** задает число значащих цифр только для данного вызова этой функции. Для глобальной установки применяется функция **digits(), во втором аргументе которой указывается требуемое количество цифр.**

Функция **digits()** служит для установки числа цифр в числах арифметики произвольной точности (по умолчанию 32).

|  |
| --- |
|  |
| **>>digits**  **Digits = 32**  **>>vpa(pi)**  **ans =**  **3.1415926535897932384626433832795**  **>>digits(5)**  **>> pi**  **ans =**  **3.1416**  **>>vpa(pi,3)**  **ans =**  **3.14**  **>>** |

Для проведения вычислений в арифметике произвольной точности служит функция ***vpa*:**

* **R = vpa(S)** — возвращает результат вычислений каждого элемента символьного массива **S**, используя арифметику произвольной точ­ности с текущим числом цифр **D**, установленным функцией **digits**. Результат R имеет тип *sym*.
* **vpa(S,D)** — возвращает результат вычислений каждого элемента массива **S**, используя арифметику произвольной точности с коли­чеством знаков чисел **D**.

|  |
| --- |
|  |
| **>>vpa(exp(1),50)**  **ans =**  **2.7182818284590450907955982984276488423347473144531**  **>>** |

Визулизация функции одной переменой осуществляется при помощи функции ezplot().

### 3.3.3. Упрощение выражений, подстановки и разложение на множетели

1. **Упращение выражений – simple()** и **simplify(S)**

Для упрощения выражений в MatLaвобщего видаиспользуются функции **simple()** и **simplify(S).** Функция**simplify(S)** реализует мощный алгоритм упрощениясимвольных выражений массива **S**, содержащих как тригонометрические, экспоненциальные и логарифмические функции, так и специальные. Алгоритм, заложенный в **simplе(S),** пытается получить выражение, которое представляется меньшим числом символов, чем исходное.Если упрощение невозможно, то возвращается исходное выражение. Например:

|  |
| --- |
|  |
| >>syms a b x  >>V = [sin(х)^2 + соs(х)^2 log(a\*b)];  >>simplify(V)  ans =  [ 1, log(a\*b)]  >>simplify((a^2 - 2\*a\*b + b^2) / (a - b))  ans =  a – b  >> |

1. **Расширение выражений – expand(S)**

Для расширения (раскрытия скобок) выражений используется функция **expand(S).** Эта функция расширяет выражения, входящие в массив **S**. Ра­циональные выражения она раскладывает на простые дроби, поли­номы — на полиномиальные разложения и т. д. Функция работает со многими алгебраическими и тригонометрическими функциями. Например:

|  |
| --- |
|  |
| >>syms a b x  >>S=[(x + 2)\*(x + 3)\*(x + 4) sin(2\*x)];  >>expand(S)  ans =  [ x^3 + 9\*x^2 + 26\*x + 24 , 2\*sin(x)\*cos(x)]  >>expand(sin(a + b))  ans =  sin(a)\*cos(b) + cos(a)\*sin(b)  >>expand((a + b)^:3)  ans =  а^3 + 3\*а^2\*Ь + 3\*а\*Ь^2 + Ь^3  >> |

1. Операция подстановки **– subs( )**

Одной из самых эффектных и часто используемых операций символьной математики является операция подстановки. Она реализуется функцией **subs( )**, имеющей несколько различных форм записи:

* **subs(S)** *–* заменяет в символьном выражении **S** все переменные их символьными значениями, которые берутся из вычисляемой функции или рабочей области системы MatLaв.
* **subs(S,NEW)**  *–* заменяет все свободные символьные переменные в **S** из списка **NEW**.
* **subs (S, OLD, NEW**) *–* заменяет **OLD** на **NEW** в символьном выражении **S**. При одинаковых размеров массивов **OLD** и **NEW** замена идет поэле­ментно. Если **S OLD** - скаляры, a **NEW** *–* числовой массив или массив ячеек, то скаляры расширяются до массива результатов.

|  |
| --- |
|  |
| >> syms a b x у  >>subs(x - y, y, l)  ans =  x - l  >> subs(sin(x) + cos(y), [x,y], [a,b])  ans =  sin(a)+cos(b)  >> |

1. **Обращение функции – finverse**

Часто возникает необходимость в задании функции, обратной по от­ношению к заданной функции **f** . Для этого в Symbolic имеется две формы функ­ции обращения inverse:

* g = finverse(f) *–* возвращает функцию, обратную к f. Считается, что f *–* функция одной переменной, например 'х'. Тогда g(f(x)) = х.
* g = finverse(f,v) *–* возвращает функцию, обратную к f, относитель­но заданной переменной v, так что g(f(v)) = v. Эта форма исполь­зуется, если f *–* функция нескольких переменных.

|  |
| --- |
|  |
| >> sym х;  >> finverse(sinh(x))  ans =  asnnh(x)  >> finverse(exp(x))  ans =  log(x) |

1. **Суперпозиция функций – compose( )**

* compose(f, g) — возвращает f(g(y))*,* где f = f(x) и g = g(y)*.* Незави­симые переменные х и у находятся с помощью функции findsym( ).

|  |
| --- |
|  |
|  |

1. **Функция разложения в ряд – taylor используется функция** taylor **( )**

**Для разложения в ряд Тейлора**

* taylor (f) – возвращает шесть членов ряда Маклорена функции f ;
* taylor (f, n, x, a) – возвращает *n* членов ряда Тейлора в точке х = а ;
* taylor (f, n) – возвращает *( n - 1)* членов ряда Маклорена функции f ;
* taylor (f, a) – возвращает шесть членов ряда Тейлора функции f в точке а.

|  |
| --- |
|  |
| **>>x = sym('x');**  **>>taylor(sin(x))**  **ans =**  **x – 1/6\*x^3 + 1/120\*x^5**  **>>taylor(int(sin(x))**  **ans =**  **- 1 +1/2\*x^2 – 1/24\*x^4**  **>>** |

1. **Разложение на множители**

Для разложения выражения на простые множители используется функция **factor(S).** Эта функция поэлементно разлагает выражения вектора **S** на простые множители, а целые числа - на произведение простых чи­сел. Следующие примеры иллюстрируют применение функции:

|  |
| --- |
|  |
| **>> x=sym('х');**  **>>factor(x^7-l)**  **ans =**  **( х – 1 )\*( х^6 + х^5 + х^4 + х^3 + х^2 + х + 1 )**  **>>factor(х^2 – х - 1)**  **ans =**  **хА2 – х - 1**  **>>factor(sym('123456789'))**  **ans =**  **(3 )^2\*(3803)\*(3607)**  **>>** |

1. **Комплектование по степеням – collect( )**

Функция co11ect(S,v) обеспечивает комплектование выражений в со­ставе вектора или матрицы S по степеням переменной v.

1. **Упрощение выражений – simple( )**

Функция **simple(S)** выполняет различные упрощения для элементов массива S и выводит как промежуточные результаты, так и самый ко­роткий конечный результат. В другой форме — [R, HOW] = **simple(S)** – промежуточные результаты не выводятся.

1. **Приведение к рациональной форме – numden**

Функция [N,D] = numden(A) преобразует каждый элемент массива А в рациональную форму в виде отношения двух неприводимых полиномов с целочисленными коэффициентами. При этом N и D — числители и знаменатели каждого преобразованного элемента массива.

|  |
| --- |
|  |
| **>> [n,d] = numden(sym(8/10))**  **n =**  **4**  **d =**  **5**  **>> syms x y**  **>> [ n, d ] = numden (x\*y + y / x)**  **n =**  **y\*(x^2 + 1)**  **d =**  **x**  **>>** |

### 3.3.4. Вычисление сумм рядов и произведений

Функция вычисления суммы рядов – **symsum**

Для аналитического вычисления суммы ряда служит команда **symsum**:

* **symsum(S)** - возвращает символьное значение суммы бесконечно­го ряда по переменной, найденной автоматически с помощью функ­ции **findsym**;
* **symsum(S, v)** - возвращает сумму бесконечного ряда по переменной **v**;
* **symsum(S, a, b)** и **symsum(S, v, a, b)** - возвращают конечную сумму ря­да в пределах номеров слагаемых от **а** до **b**.

**Пример 3.3-1.**

|  |
| --- |
| **Пример 3.3-1.** |
| **>> x = sym(‘x’);**  **>> symsum(x^2)**  **ans =**  **1/3\*x^3-1/2\*x^2+1/6\*x**  **>>symsum([x, x^2, x^3], 1, 5)**  **ans =**  **[ 15, 55, 225]**  **>>** |

### 3.3.5. Вычисление пределов

Напомним, что число **L** называется пределом функции **f(x**) в точке **а**, если при **х**, стремящимся к **а** (или **х → а**), значение функции неограниченно приближается к **L**. Это обозначается следующим образом: **lim f(x) = L**.

Предел может быть конечным числом, положительной или отрицательной бесконечностью.

Если функции (например, разрывные в точке **x=a**), у которых нет предела в самой точке **x=a**, но есть предел при **x → a – 0** или при **x→ a + 0**, где под **0** подразумевается очень малое число. В первом случае говорят о существовании предела слева от точки **x=a**, а во втором – справа от этой точки. Если эти пределы равны, то существует предел функции в точке **x=a**.

Для вычисления пределов аналитически заданной функции **f(x)** служит функция **limit(),** котораяможет записываться в нескольких вариантах:

* **limit(f, x, a)** – возвращает предел символьного выражения **f** в точке **х → а**;
* **limit(f, a)** – возвращает предел для независимой переменной, определяемой **findsym()**;
* **limit(f)** - возвращает предел при **a=0**;
* **limit(f, x, a, `right`)** или **limit(f, x, a, `left`)** – возвращает предел в точке **а** справа или слева.

**Пример 3.3-2.**

|  |
| --- |
| **Пример 3.3-2.** |
| **>> syms a x**  **>> limit(sin(a\*x)/(a\*x))**  **ans =**  **1**  **>> limit(sin(a\*x)/x)**  **ans =**  **a**  **>> limit(2\*sin(x)/x)**  **ans =**  **2**  **>> limit(2+sin(x)/x,0)**  **ans =**  **3**  **>> limit(tan(x),pi)**  **ans =**  **0**  **>> limit(tan(x),pi/2)**  **ans =**  **NaN**  **>> limit(tan(x),x,pi/2,'right')**  **ans =**  **-Inf**  **>> limit(tan(x),x,pi/2,'left')**  **ans =**  **Inf**  **>>** |

**Пример 3.3-3**.  Доказать непрерывность функции в точке **y=x2**в точке **x=2**.

Для доказательства непрерывности необходимо вычислить предел и значение функции в этой точке.

|  |
| --- |
| **Пример 3.3-3** |
| **>> sym x;**  **>> f=sym('x^3');**  **>> limit(f,x,2,'left')**  **ans =**  **8**  **>>** |

### 3.3.6. Вычисление производной в символьном виде и производной в заданной точке

Для вычисления в символьном виде производных от выражения **S** служит функция **diff()**, записываемая в формате

**diff(S,'v')** или **diff(S, sym('v'))**.

Она возвращает символьное значение первой про­изводной от символьного выражения или массива символьных вы­ражений **S**  по переменной **v**. Эта функция возвращает .

Для вычисления n –й производной используется следующие форматы:

* **diff(S, n)** – возвращает **n***-*ю *(***n**–целое число) производную символьного выражения или массива символьных выражений **S**по переменной **v**.
* **diff(S, v, n)** и **diff(S, n, v)** – возвращает **n**-юпроизводную **S** по переменной **v**.

**Пример 3.3-4.**

|  |
| --- |
| **Пример 3.4-4.** |
| **>>x =sym( 'x' ); y=sym( 'у' );**  **>> diff(x^y)**  **ans =**  **х^у\*y / x**  **>>slmplify(ans)**  **ans =**  **x^(y-1)\*y**  **>>diff(s1n(y\*x), x, 3)**  **ans =**  **- cos(y\*x)\*y^3**  **>>diff([x^3 sin(x) exp(x)], x)**  **ans =**  **[ 3\*x^2, cos(x), exp(x)]**  **>>**  **syms x**  **f = sin(5\*x)**  **ans =**  **5\*cos(5\*x)**  **g = exp(x)\*cos(x)**  **diff(g)**  **ans =**  **exp(x)\*cos(x)-exp(x)\*sin(x)**  **To take the second derivative of g, enter**  **diff(g,2)**  **ans =**  **-2\*exp(x)\*sin(x)**  **You can get the same result by taking the derivative twice:**  **diff(diff(g))**  **ans =**  **-2\*exp(x)\*sin(x)**  **c = sym('5');**  **diff(c)**  **ans =**  **0**  **diff(5)**  **Так 5 не символьная переменная** |

**Пример 3.3-5.**

|  |
| --- |
| **Пример 3.4-5.** |
| **sym(f,1)**  **ans =**  **t**  **diff(f,t,2)**  **ans =**  **-sin(s\*t)\*s^2**  **syms a b x n t theta**  **f =x^n;**  **diff(f)**  **ans=**  **x^n\*n/x**  **f=sin(a\*t+b);**  **diff(f)**  **ans=**  **cos(a\*t+b)\*a**  **f=exp(i\*theta);**  **diff(f)**  **ans=**  **i\*exp(i\*theta)** |

**Пример 3.3-6.**

|  |
| --- |
| **Пример 3.4-6.** |
| **syms a x**  **A = [cos(a\*x),sin(a\*x);-sin(a\*x),cos(a\*x)]**  **A =**  **[cos(a\*x), sin(a\*x)]**  **[-sin(a\*x), cos(a\*x)]**  **diff(A)**  **ans =**  **[-sin(a\*x)\*a, cos(a\*x)\*a]**  **[-cos(a\*x)\*a, -sin(a\*x)\*a]** |