Символьные вычисления в среде MATLAB

1.1. Объявление символьных переменных и констант

В процессе символьных вычислений используются переменные и константы особого типа, так называемые символьные объекты. Хотя обычно в коде MATLAB тип переменных определяется динамически и нет нужды объявлять его явно, для символьных объектов дело обстоит иначе. Для объявления символьных переменных служит команда **syms**, которая в качестве аргументов принимает имена переменных, перечисленные
через пробел. Например, так:

```
>> syms x y
>> syms a b real
% объявляемые объекты обозначают вещественные переменные
```

Объявление символьных констант осуществляется при помощи функции **sym**. Она может принимать в качестве аргумента строку, содержащую специальные переменные, численное выражение или вызов функции, как в примерах ниже:

```
>> sym_pi = sym('pi')
>> sym_delta = sym('1/10')
>> sym_sqroot2 = sym('sqrt(2)')
```

Использование символьных констант полезно тем, что вычисления с ними производятся точно (т.е. без вычислительных погрешностей) до тех пор, пока не потребуется вычислить некоторое числовое значение. Заметим, что при выводе содержимого рабочего пространства командой **whos** символьные переменные и константы отображаются как представители класса **sym object**.

1.2. Символьные выражения и манипуляции над ними

После объявления, с символьными переменными можно обращаться примерно так же, как и с обычными числовыми. В частности, для них определены операторы + - * / ^, с помощью которых можно составлять символьные выражения:

```
>> syms s t A
>> f = s^2 + 4*s + 5
f =
s^2 + 4*s + 5
>> g = s + 2
g =
s + 2
>> h = f*g
h =
(s^2 + 4*s + 5)*(s + 2)
>> z = exp(-s*t)
z =
exp(-s*t)
>> y = A*exp(-s*t)
y =
A*exp(-s*t)
```

Заметим, что MATLAB всегда остается прежде всего матричным процессором и потому к символьным переменным можно свободно применять матричную и векторную запись и соответствующие встроенные операторы и функции. Приведем пример:

```
>> n = 3;
>> syms x;
>> B = x.^((0:n)'*(0:n))
```

```
B =
[ 1, 1, 1, 1]
[ 1, x, x^2, x^3]
[ 1, x^2, x^4, x^6]
[ 1, x^3, x^6, x^9]
```

1.3. Интегрирование

Символьное вычисление интегралов (как определенных, так и неопределенных) выполняется одной функцией **int**:
int(S)
Интегрирует выражение S по независимым переменным, определенным функцией **findsym**.
Если S константа, то интегрирование выполняется по x: int(S, t). Интегрирует выражение S по переменной t: int(S, a, b). Вычисляет определенный интеграл на промежутке [a;b]: int(S, t, a, b).
Приведем несколько примеров:

```
>> syms x n a b t
>> int(x^n)
ans =
x^(n+1)/(n+1)
>> int(x^3 + 4*x^2 + 7*x + 10)
ans =
1/4*x^4+4/3*x^3+7/2*x^2+10*x
>> int(x, 1, t)
ans =
1/2*t^2-1/2
>> int(x^3, a, b)
ans =
1/4*b^4-1/4*a^4
>> int(cos(x))
ans =
sin(x)
>> int(exp(-x^2))
ans =
1/2*pi^(1/2)*erf(x)
```

1.4. Решение ОДУ при помощи функции dsolve

Функция **dsolve** может находить как общее решение ОДУ, так и частное решения для заданных начальных или граничных условий. При этом следует соблюдать определенные
ограничения на форму записи уравнения (или системы уравнений). Так, если неизвестная функция обозначена символической переменной y, то ее производные следует обозначать как
D[n] y, где в квадратных скобках указан порядок производной. Таким образом, производная должна обозначаться в символьном выражении Dy, вторая производная D2y и т.д. Функция dsolve может вызываться с разным набором параметров, в зависимости от типа решаемой задачи и порядка системы уравнений.
Пусть необходимо решить задачу Коши вида
$y'' + 2y' + y = x\sin 2x$; $y(0) = 1$; $y'(0) = 1$:
Опишем необходимые для этого действия.

```
clear all;                                  % Очищаем память
eq = 'D2y + 2*Dy + y - x*sin(2*x)';         % Описываем уравнение
x0 = 0;                                     % Начальные условия
y0 = 1;
y10 = -1;
ic1 = sprintf('y(%d) = %d', x0, y0);        % Преобразуем их
ic2 = sprintf('Dy(%d) = %d', x0, y10);      % в символьные выражения
fprintf('Дифференциальное уравнение: \n%s=0\n', eq);
fprintf('Начальные условия: \n%s\n%s\n', ic1, ic2);
```

```
Sol = dsolve(eq, ic1, ic2, 'x');                    % Решаем начальную задачу
if isempty(Sol)                                     % Если решений нет, то ...
        disp('Решения не найдены');                 % печатаем ответ
else                                                % В противном случае ...
        n = length(Sol);
        fprintf('Найдено решений: %d\n', n)
        for i = 1:n                                 % печатаем все решения
                fprintf('Решение %d: \ny=%s\n', i, char(Sol(i)));
end
end
```