Практическая работа 9.
Решение систем обыкновенных дифференциальных уравнений в системе MatLab

Цель работы: приобретение навыков решения систем обыкновенных дифференциальных уравнений с использованием специальных функций-решателей пакета MATLAB.

Теоретические сведения

Решение обыкновенных дифференциальных уравнений
Метод Рунге-Кутта

Будем рассматривать численные методы решения задачи Коши. Пусть на отрезке $a \le x \le b$ требуется найти решение дифференциального уравнения

$$y' = f(x, y), \tag{1}$$

удовлетворяющее начальному условию

$$y(a) = y_0. \tag{2}$$

Предполагается, что условия существования и единственности решения задачи Коши выполнены. На практике найти решение задачи Коши удается не всегда, поэтому используют приближенные методы решения. Отрезок $[a, b]$ разбивается на интервалы с шагом h (как правило, постоянным), полагают $h = x_{n+1} - x_n$, и находят значение $y_{n+1} = y(x_{n+1})$. При этом получают таблицу, состоящую из вектора аргументов $x = (x_0, x_1, \ldots, x_n)$ и соответствующего ему вектора функции $y = (y_0, y_1, \ldots, y_n)$.

Из общего курса обыкновенных дифференциальных уравнений известен аналитический метод, основанный на использовании ряда Тейлора. При этом приближенное решение исходной задачи ищут в виде

$$y_m(x_{n+1}) \approx \sum_{i=0}^{m} \frac{h^i}{i!} y^{(i)}(x_n), \tag{3}$$

где $x_{n+1} = x_n + h$, $y^{(0)}(x_0) = y(x_0)$, $y^{(1)}(x_0) = y'(x_0) = f(x_0, y_0)$, а значения $y^{(i)}(x_0)$ при $i = 2, 3, \ldots, m$ находят по формулам, полученным последовательным дифференцированием уравнения (1). В случае $m = 1$ приближенное равенство (3) не требует вычисления производных правой части уравнения и позволяет с погрешностью порядка h^2 находить значение $y(x_n)$. Соответствующая формула имеет вид:

$$y_{n+1} = y_n + h \cdot f_n \tag{4}$$

и носит название формулы Эйлера.

Ниже приводятся формулы Рунге-Кутта для решения уравнения (1):

$$y_{i+1} = y_i + \Delta y_i, \tag{5}$$

где $\Delta y_i = \dfrac{k_1 + 2k_2 + 2k_3 + k_4}{6}$,

$$k_1 = h \cdot f(x_i, y_i),\ k_2 = h \cdot f(x_i + \frac{h}{2}, y_i + \frac{k_1}{2}),\ k_3 = h \cdot f(x_i + \frac{h}{2}, y_i + \frac{k_2}{2}),$$

$$k_4 = h \cdot f(x_i + h, y_i + k_3),\ i = 0, 1, 2, \ldots, m.$$

Метод Рунге-Кутта имеет порядок точности h^4. Приближенно оценку погрешности метода Рунге-Кутта можно вычислить по формуле:

$$\varepsilon = \frac{|y_{2h} - y_h|}{15}, \tag{6}$$

где y_{2h} и y_h - результаты вычислений по схеме (5) с шагом h и шагом $2h$. Метод Рунге-Кутта применим также для решения системы дифференциальных уравнений вида:

$$\begin{cases} y' = f(x, y, z) \\ z' = \varphi(x, y, z) \end{cases}$$

с заданными начальными условиями: $y = y_0$, $z = z_0$ при $x = x_0$.

Для решения задачи (6.1) по методу Милна, исходя из начальных условий $y = y_0$ при $x = x_0$, находим каким-либо способом последовательные значения

$$y_1 = y(x_1),\ y_2 = y(x_2),\ y_3 = y(x_3)$$

исходной функции $y(x)$ (например, применить метод Рунге-Кутта). Приближения

y_i и $\bar{y}_i$ для следующих значений y_i ($i = 4, 5, 6, \ldots, n$) последовательно находятся по формулам:

$$y_i = y_{i-2} + \frac{h}{3}(\bar{f}_i + 4f_{i-1} + f_{i-2}), \tag{7}$$

где $\bar{y}_i = y_{i-4} + \dfrac{4h}{3}(2f_{i-3} - f_{i-2} + 2f_{i-1})$,

$$f_i = f(x_i, y_i),\ \bar{f}_i = f(x_i, \bar{y}_i).$$

Полагая в формуле (7) $i = 3$, находим последовательно $\bar{y}_4$, $\bar{f}_4, y_4$. Затем, полагая $i = 4$, находим $\bar{y}_5$, $\bar{f}_5, y_5$ и т.д. Найденные значения y_4, y_5, y_6, … и являются приближенными значениями решения $y(x)$ при $x = x_4$, x_5, x_6, …, где $x_i = x_0 + ih$. Погрешность, получающуюся при вычислении y_i по данному методу, можно вычислить по формуле:

$$\varepsilon_i = \frac{1}{29}\left|\bar{y}_i - y_i\right|.$$

Поэтому при вычисления можно попутно проверять, не выходит ли эта погрешность за пределы принятой степени точности вычислений. Если это где-либо произойдет, то надо уменьшить шаг. Отметим, что суммарная ошибка данного метода имеет порядок h^4.

Одношаговые методы решения задачи Коши имеют один существенный недостаток, который заключается в том, что при построении этих методов требуется информация о решаемой задаче на отрезке длиной в один шаг. Поэтому эта информация на каждом этапе процесса должна быть заново получена и, соответственно, это вызывает большую трудоемкость вычислительных процедур. Можно построить вычислительные методы таким образом, чтобы информация о решаемой задаче могла быть использована на нескольких шагах вычислительного процесса. Такие методы получили название многошаговых методов.

Метод Адамса

Метод Адамса позволяет получить более точные результаты, чем метод Эйлера и его уточнения. В основе метода Адамса лежит известное соотношения между функцией и ее производной:

$$\Delta y_k = y_{k+1} - y_k = \int_{x_k}^{x_{k+1}} y'(x)dx \tag{8}$$

Обозначим через $q(x) = hy'(x)$ и заменим $q(x)$ интерполяционным полиномом $t(x)$, построенным по значениям $t(x)$ в предыдущих точках. Степень полинома r выбирается из условия постоянства конечных разностей r-го порядка. Обычно уже разности третьего порядка практически постоянны.

В этом случае, применяя вторую интерполяционную формулу Ньютона, можно на отрезке $[x_{k-3}, x_k]$ с узлами интерполяции $x_{k-3}, x_{k-2}, x_{k-1}, x_k$ построить интерполяционный полином::

$$q(x) = q(x_k + th) = q_k + t\Delta q_{k-1} + \frac{t(t+1)}{2!}\Delta^2 q_{k-2} + \frac{t(t+1)(t+3)}{3!}\Delta^3 q_{k-3}, \tag{9}$$

где

$$t = \frac{x - x_k}{h} \tag{10}$$

Преобразуем формулу (8), произведя замену переменной на основе (9):

$$x = x_k + th$$

$$\Delta y_k = \int_{x_k}^{x_{k+1}} y'(x)dx = \int_0^1 q(x_k + th)dt$$

Подставив эту формулу в выражение для $q(x + th)$ и произведя интегрирование, окончательно получим:

$$\Delta y_k = q_k + \frac{1}{2}\Delta q_{k-1} + \frac{5}{12}\Delta^2 q_{k-2} + \frac{3}{8}\Delta^3 q_{k-3} \qquad (11)$$

Эта формула называется экстраполяционной формулой Адамса, она применяется для предсказания значения $y_{k+1} = y_k + \Delta y_k$. Полученное значение Δy_k следует еще уточнить по формуле «коррекции»:

$$\Delta y_k = q_k + \frac{1}{2}\Delta q_k - \frac{1}{12}\Delta^2 q_{k-1} + \frac{1}{24}\Delta^3 q_{k-2} \qquad (12)$$

Формула (12) называется интерполяционной формулой Адамса.

Создание файлов-функций в системе MatLab

Файлы-функции являются разновидностью М-файлов. Эти файлы имеют расширение .m, а в качестве заголовка используют оператор *function*. В общем виде первая строка файла-функции может быть записана следующим образом:

function <выходные параметры>=<имя функции>(входные параметры)

Файлы-функции и файлы-сценарии нельзя отличить по расширению имени файла .m. Процедуры, предназначенные на многократное использование файлами-сценариями, оформляются в виде файлов-функций. Переменные, используемые в рамках файлов-функций, являются локальными переменными. Область, используемая локальными переменными, после выполнения файла-функции освобождается для других переменных. Переменные единого рабочего пространства с помощью оператора *global* могут быть объявлены в качестве глобальных переменных. В этом случае функция может обратиться к указанным переменным.

Необходимо, чтобы имя М-файла, в котором записывается программа вычислительного процесса, совпадало с именем функции. Имя функции не должно превышать 31 символ. Имя может быть и длиннее, но система MatLab принимает во внимание только первые 31 символ. Имя файла-функции должно начинаться с буквы, остальные символы могут быть любой комбинации букв, цифр и подчеркиваний.

Пример 1. Написать функцию для вычисления факториала. Файл должен быть сохранен с именем *factorial.m*.

```
function y = factorial(n)
k = 1;
for i = 1:n
    k = k * i;
end
y = k;
```

Для вычисления 5! Достаточно набрать в текущей строке команду:

>> k = factorial(5)

k =

120

*Структура файла-функции***.** Файл-функция состоит из строки определения функции, первой строки комментария, собственно комментария, тела функции, строчных комментариев.

Строка определения функции сообщает системе MatLab, что процедура является файлом-функцией, а также определяет список входных аргументов. Если функция имеет более одного выходного аргумента, то список выходных аргументов помещается в квадратные скобки. Входные аргументы, если они присутствуют, помещаются в круглые скобки. Для отделения аргументов во входном и выходном списках применяются запятые.

Комментарий. Для М-файлов можно создать online-подсказку, вводя текст в одной или более строках. При вводе команды подсказки *help <имя_функции>*, система MatLab отображает строки комментария, которые размещаются между строкой определения функции и первой пустой строкой, либо началом программы. Команда *help <имя_функции>* игнорирует комментарии, размещенные вне этой области.

Тело функции. Тело функции содержит код языка MatLab, который выполняет вычисления и присваивает значения выходным аргументам. Операторы в теле функции могут состоять из вызовов функций, программных конструкций для управления потоком команд, интерактивного ввода/вывода, вычислений, присваиваний, комментариев и пустых строк.

Вызов функции. При вызове m-функции система MatLab транслирует функцию в псевдокод и загружает в память. Псевдокод остается в памяти до тех пор, пока не будет использована команда *clear* или завершен сеанс работы.

Существует несколько разновидностей m-функций. Функции, имена которых совпадают с именами М-файлов, называются головными функциями. Подфункции представляют собой такой вид функций, описания которых находятся в М-файле после за головной функцией. Особенность использования подфункций М-файла заключается в том, что они не могут быть вызваны извне. Эти функции предусматривают внутреннее использование. Вызов подфункций осуществляет головная функция М-файла. К вложенной функции всегда может обращаться окружающая ее функция.

Для организации ветвлений внутри выполнения вычислительной процедуры применятся условные операторы.

Ниже приведены конструкции условных операторов:

1 вариант. *if <условие>*

<операторы>

end

Операторы (тело выражения) выполняются только в том случае, если условие истинно, если условие ложно, то тело выражения не выполняется.

2 вариант. *if <условие>*

<операторы 1>

else

<операторы 2>

end

3 вариант. *if <условие1>*

<операторы 1>

elseif <условие2>

<операторы2>

elseif <условие3>

<операторы3>

. . .

else

<операторы>

end

В среде MatLab применяются следующие операторы сравнения: < – меньше; <= – меньше или равно; > – больше; >= – больше или равно; = – равно; ~= – не равно. В среде MatLab предусмотрено использование следующих логических операций: & – логическое «и» (and); | – логическое «или» (or); ~ – логическое отрицание (not). В результате выполнения логических операций получают значения 0 (false) и 1 (true).

Как правило, при разработке программ требуется использовать операторы цикла. Условный оператор цикла (или оператор цикла с предусловием) осуществляет повторение операторов нефиксированное число раз. Формат оператора имеет следующий вид:

while <условие>

<операторы>

end

Операторы выполняются, если переменная «условие» имеет ненулевые элементы.

Формат арифметического оператора цикла можно представить следующим образом:

for <имя> = <начальное значение>: <шаг>: <конечное значение>

<операторы>

end

где *<имя>* – имя переменной цикла, *<начальное значение>* – это начальное значение переменной цикла, *<конечное значение>* – это конечное значение управляющей переменной. Величина *<шаг>*

указывает значение приращения переменной цикла в процессе ее изменения от начального значения до конечного значения. Если величина шага не задана, то по умолчанию значение шага принимается равным единице.

Решение систем обыкновенных дифференциальных уравнений в системе MatLab

Для решения систем обыкновенных дифференциальных уравнений (ОДУ) в системе MatLab имеются функции: *ode23, ode45, odell3, odel5s, ode23s, ode23t* и *ode23tb*.

Наиболее употребительной является функция *ode45*, реализующая алгоритм Рунге - Кутта 4–5-го порядка (разные порядки точности используются для контроля шага интегрирования).

Пусть необходимо решить систему *n* дифференциальных уравнений, разрешенных относительно первых производных функций $y_1, y_2, ..., y_n$:

$y_1' = F_1(t, y_1, y_2, ..., y_n);$

$y_2' = F_2(t, y_1, y_2, ..., y_n);$

...

$y_n' = F_n(t, y_1, y_2, ..., y_n).$

Введем вектор-столбцы Y и F, состоящие из $y_1, y_2, ..., y_n$ и $F_1, F_2, ..., F_n$, соответственно. Тогда система дифференциальных уравнений примет следующий векторный вид:

$Y' = F(t, Y).$

Решатель обыкновенных дифференциальных уравнений обеспечивает возможность выбора метода, задания начальных условий и др.

[T, Y] = solver('F', [DT], Y0, ...)

где *DT* – диапазон, содержащий начальное и конечное значение аргумента, *Y0* – вектор начальных значений переменных состояния, *F* – имя функции вычисления правых частей системы обыкновенных дифференциальных уравнений, solver – имя используемой функции (*ode45* – метод Рунге-Кутта 4 и 5-го порядков, *ode23* – тот же метод 2 и 3-го порядков, *ode113* – метод Адамса для нежестких систем, *ode23s* и *ode15s* – для жестких систем и др.). Версии решателя различаются используемыми методами и временем решения.

Под жесткостью принято понимать повышенное требование к точности – использование минимального шага во всей области интегрирования. При отсутствии информации о жесткости рекомендуется использовать решение с помощью *ode45* или *ode15s*.

Чтобы применить «решатель» *ode45*, нужно оформить в виде функции пользователя правую часть системы уравнений *F(t, Y)*.

Ниже приведен пример использования М – файлов для решения систем обыкновенных дифференциальных уравнений.

Пример 2. Решить систему дифференциальных уравнений с заданными начальными условиями. Построить график.

Решение: определяем систему дифференциальных уравнений в файле *func1.m*

```
function f=func1(t,a)
f = [-a(1)+ 6;
a(1)-4*a(2);
2*a(2)-3*a(3)];
```

Далее определяем:

```
>> x1=[0,10,0];
>> [T,Y]=ode45('func1',[0,2],x1);
```

И выполняем построение графиков.