

Обыкновенные дифференциальные уравнения

С помощью дифференциальных уравнений можно описать разные задачи: движения системы, взаимодействующих материальных точек, химической кинетики и т.д. Различают три типа задач для систем диф. уравнений:

- Задача Коши
- Краевая задача
- Задача на собственные значения

Кратко расскажем о их сути:

Задача Коши предполагает дополнительные условия в виде значения функции в определённой точке.

Краевая задача подразумевает поиск решения на заданном отрезке с краевыми (граничными) условиями в концах интервала или на границе области.

Задача на собственные значения — помимо искоемых функций и их производных, в уравнение входят дополнительное несколько неизвестных параметров, которые являются собственными значениями.

Методы решения дифференциальных уравнений

Решение ОДУ в Matlab и не только, в первую очередь, сводится к выбору порядка численного метода решения. Порядок численного метода не связан с порядком дифференциального уравнения. Высокий порядок у численного метода означает его скорость сходимости.

В случае большого интервала, с помощью алгоритмов с низким порядком сжимают интервал с решениями и находят приблизительные корни, а затем уже уточняют корни с помощью методов с высоким порядком.

Решение обыкновенных дифференциальных уравнений в Matlab можно реализовать «своими ручками», прописав алгоритм по разным схемам. Но также в Matlab есть встроенные функции, выполняющие все стандартные задачи.

Метод Рунге-Кутты первого порядка

$$y_{i+1} = y_i + h * f(x_i, y_i)$$

Методы Рунге-Кутты представляют собой разложения в ряд Тейлора и от количества использованных элементов ряда зависит порядок этого метода.

Следовательно, помимо Рунге-Кутты первого порядка, вы сможете увидеть методы других порядков. Иногда их называют другими именами.

Метод Рунге-Кутта второго порядка

$$y_{i+1} = y_i + (h/2) * f(x_i, y_i) + (h/2) * f(x_{i+1}, y_{i+1})$$

Также известен как **Метод Эйлера-Коши**. Как видите, во второй части уравнения происходит обращения к следующему шагу. Но как тогда быть, если нам ещё не известен следующий шаг? Всё просто. Метод Рунге-Кутта второго порядка — это всё тот же метод первого порядка, однако, на половине шага происходит нахождение «первичного» решения, а затем происходит его уточнение. Это позволяет поднять порядок скорости сходимости до двух.

Мы не будем говорить о третьем порядке, потому что задачи на третий порядок встречаются редко, но если будет необходимо, пишите в комментариях, выложу.

Метод Рунге-Кутта четвёртого порядка

$$\begin{aligned} T_1 &= h * f(x_i, y_i) \\ T_2 &= h * f(x_i + h/2, y_i + T_1/2) \\ T_3 &= h * f(x_i + h/2, y_i + T_2/2) \\ T_4 &= h * f(x_i + h/2, y_i + T_3) \\ y_{i+1} &= y_i + (T_1 + 2 * T_2 + 2 * T_3 + T_4) / 6 \end{aligned}$$

Метод Рунге-Кутта четвёртого порядка считается самым распространённым. Тем не менее, работает он аналогично второму и третьему порядку.

Решение ОДУ в Matlab стандартными средствами

Стоит отметить, что мы с вами разобрали только один самый известный метод решения ОДУ с разными порядками. Однако, методов очень много.

Для решения дифференциальных уравнений и систем в MATLAB предусмотрены следующие функции:

ode45 (f, interval, X0, [options])
ode23 (f, interval, X0, [options])
ode113 (f, interval, X0, [options])
ode15s (f, interval, X0, [options])
ode23s (f, interval, X0, [options])
ode23t (f, interval, X0, [options])
ode23tb (f, interval, X0, [options])

Входными параметрами этих функций являются:

- **f** — вектор-функция для вычисления правой части уравнения системы уравнений;
- **interval** — массив из двух чисел, определяющий интервал интегрирования дифференциального уравнения или системы;
- **X0** — вектор начальных условий системы дифференциальных уравнений;
- **options** — параметры управления ходом решения дифференциального уравнения или системы.

Все функции возвращают:

- массив **T** — координаты узлов сетки, в которых ищется решение;
- матрицу **X**, *i*-й столбец которой является значением вектор-функции решения в узле T_i .

В функции **ode45** реализован метод Рунге-Кутты (Кэша-Кара) 4-5 порядка точности, в функции **ode23** также реализован метод Рунге-Кутты, но 2-3 порядка, а функция **ode113** реализует метод Адамса.

Для решения жёстких систем предназначены функция **ode15s**, в которой реализован метод Гира, и функция **ode23s**, реализующая метод Розенброка. Для получения более точного решения жёсткой системы лучше использовать функцию **ode15s**. Для решения системы с небольшим числом жёсткости можно использовать функцию **ode23t**, а для грубой оценки подобных систем служит функция **ode23tb**.

Символьное решение обыкновенных дифференциальных уравнений произвольного порядка осуществляет функция **dsolve** $r = \text{dsolve}('eq1,eq2,...', 'cond1,cond2,...', 'v')$

Пример использования:

```
% в отдельном М-файле с именем pr10
function f=pr10(x,y)
f=sin(x+y)+(3/2)*(x-y);
end)

%в основном М-файле
ode113(@pr10,[0 20],0) %Метод Адамса: @pr10 - ссылка на М-функцию,
% [0 20]- интервал, 0 - условие: y(0)=0
%пример (u'=u^2)
expl=dsolve('Du=u^2','x')
```