Численное решение обыкновенных дифференциальных уравнений. Задача Коши и краевая задача.

Шаго Павел Евгеньевич

Санкт-Петербург 2024

Дифференциальные уравнения — один из главных инструментов математического моделирования физических и технических объектов.

Рассмотрим самый простой вариант, диффур 1 порядка разрешенный относительно производной

$$dy/dt = f(t,y) \quad (1)$$

И вариант посложнее

$$F(t, y, dy/dt) = 0 \quad (2)$$

Аналитически преобразовать (2) в (1) удается не всегда, но вот численно когда нас интересуют только численные значения производной преобразовать (2) в (1) легко.

Если y — вектор, то имеем дело с системой ду.

ДУ содержащие производные высших порядков обычно сводят к

системам уравнений 1 порядка с большим числом неизвестных.

$$d^2y/dt^2 = F(t, y, dy/dt) \ \{y = y_1, dy_1/dt = y_2\}$$

$$y_1' = y_2$$

$$y_2' = F(t, y_1, y_2)$$

Задача Коши

Отыскать функцию $y = y(t)$ удовлетворяющую начальным условиям $y(t_0) = y_0$

Способы численного интегрирования ду разрабатывались начиная с Эйлера

который предложил метод ломанных где заменяем $dy/dt = \Delta y/\Delta t$

$$t_{k+1} = t_k + h$$

$$y_{k+1} = y_k + hf(t_k, y_k)$$

где $h = \Delta t$ может быть как постоянным так и переменным

А в неявном методе Эйлера

$$t_{k+1} = t_k + h$$

$$y_{k+1} = y_k + hf(t_{k+1}, y_{k+1})$$

Для нахождения y_{k+1} необходимо решить уравнение относительно этой переменной

Метод итерации

$$y_{r+1}^{(0)} = y_k,\ y_{r+1}^{(1)} = y_k + hf(t_{k+1},\ y_{r+1}^{(0)}),\ \ldots$$

Условие сходимости $h\left\|\frac{\partial f}{\partial y}\right\| < 1$

Линейная система относительно Δy (метод Розенброка)

$$(I - h(\partial f/\partial y))\Delta y = hf$$

Формула трапеций

$$t_{k+1} = t_k + h; \qquad y_{k+1} = y_k + (f(t_k,\ y_k) + f(t_{k+1},\ y_{k+1}))h/2$$

Формулы Рунге-Кутты 5-ого порядка:

$$k_1 = hf(t_k, y_k);$$
$$k_2 = hf(t_k + h/2, y_k + k_1/2);$$
$$k_3 = hf(t_k + h/2, y_k + k_2/2);$$
$$k_4 = hf(t_k + h, y_k + k_3);$$
$$y_{k+1} = y_k + (k_1 + 2k_2 + 2k_3 + k_4)/6$$

Явный метод Адамса 2-го порядка:

$$y_{k+1} = y_k + (3f_k - f_{k-1})h/2$$

Неявный метод Адамса 3-го порядка:

$$y_{k+1} = y_k + (5f_{k+1} + 8f_k - f_{k-1})h/12$$

“Дифференцирование назад” по Куртису и Хиршфельдеру:

$$y_{k+1} = (4y_k + y_{k-1} + 2hf_{k-1})/3$$

В MATLAB есть семейство функций ode{xxx} для решения ЗК

ode45 — явный метод Рунге-Кутты 4 и 5 порядка

ode23 — явный метод Рунге-Кутты 2 и 3 порядка

ode113 — метод Адамса, Башфорта и Моултона (типа предиктор-корректор) переменного порядка

ode15s — “дифференцирование назад” и численные методы дифференцирования

ode23s — модифицированный метод Розенброка 2-го порядка

ode23t — использует метод трапеций

[tout, yout] = ode{xxx}(fun, tspan, y_0)

В вектор tspan можно задать “контрольные значения“

Если tspan задан, то tout = tspan

Дифференциальное уравнение y"=-g описывает движение материальной точки в гравитационном поле Земли (константа g=9.8). Обозначив y1=y, y2=y', получим систему дифференциальных уравнений:

$y_1' = y_2$

$y_2' = -g$

Сформируем вектор-столбец начальных значений

```
y0=[0; 10]
```

(начальная координата 0, начальная скорость 10) и вектор контрольных значений

```
tspan=0:0.2:2
```

Функцию вычисления правых частей в нашем случае можно оформить в виде анонимной функции:

```
>> dydt=@(t,y)[y(2); -9.8];
```

Обратимся к функции ode45:

```
>> [tout,yout] = ode45(dydt,tspan,y0)
```

Результаты представлены в табл. 14.3.

Таблица 14.3

tout	yout(1)	yout(2)	tout	yout(1)	yout(2)
0	0	10.0000	1.2000	4.9440	-1.7600
0.2000	1.8040	8.0400	1.4000	4.3960	-3.7200
0.4000	3.2160	6.0800	1.6000	3.4560	-5.6800
0.6000	4.2360	4.1200	1.8000	2.1240	-7.6400
0.8000	4.8640	2.1600	2.0000	0.4000	-9.6000
1.0000	5.1000	0.2000			

Но также можно осуществлять дополнительный контроль над этими функциями

[tout, yout, varargout] = ode{xxx}(fun, tspan, y_0, options, varargin)

Параметры options задаются обращением к функции odeset ⟸ <’var’, val>

Примеры параметров:

RelTol — допустимая относительная погрешность вычислений (1e-3)

AbsTol — -||- абсолютная -||- (1e-6)

InitialStep — начальный шаг (автоматически)

Mass — определяет матрицу масс M(t, y)y’ = f(t, y)

Events — некоторое событие описанное как функция в .m файле при наступлении которого ход вычислений должен каким-то образом поменяться

Jacobian — задает в аналитическом виде $\partial f/\partial y$ (15s, 23s, 23t(b))

Краевая задача для обыкновенных дифференциальных уравнений заключается в отыскании функции y = y(x), которая на отрезке [a, b] удовлетворяет этому уравнению и граничным условиям наложенным на значения функции и/или её производной на концах отрезка в (a, b)

Для одного уравнения первого порядка КЗ некорректна, т. к. значение y(a) полностью определяет поведение функции на всем отрезке в том числе и значение y(b)

Для уравнения y'' = f(x, y, y') граничные условия могут выглядеть как (A, B, α_0, α_1, β_0, β_1)

$$y(a) = A,\ y(b) = B;$$

$$\alpha_0 y(a) + \alpha_1 y'(a) = A,\ \beta_0 y(b) + \beta_1 y'(b) = B$$

Функция bvp4c решает краевую задачу для системы обыкновенных ду

начальные значения задаются функцией bvpinit

solinit = bvpinit(xinit, yinit)

sol = bvp4c(odefun, bcfun, solinit)

Для построения графиков необходимо воспользоваться вспомогательной функцией bvpval

yy = bvpval(sol, xx), где xx — вектор пробных точек

При полном обращении можно также устанавливать дополнительные параметры

options и дополнительные аргументы для вычисления odefun и bcfun

через функцию bvpset ⟸ <’var’, val>

Пусть y’’ + y = 0, y(0) = 0, y(pi) = 1

заменим системой $y_1' = y_2$, $y_2 = -y_1$

```
xinit=linspace(0,pi,5)
xinit =
         0    0.7854    1.5708    2.3562    3.1416
```

```
function dydx = exampl(x,y)
dydx = [y(2) -y(1)];
```

```
function res = border(ya,yb)
res = [ ya(1) yb(1)-1];
```

```
>> solinit = bvpinit(xinit,yinit)
solinit.x =
    0 0.7854 1.5708 2.3562 3.1416
```

```
solinit.y =
     0     0     0     0     0
     1     1     1     1     1
```

```
>> solinit = bvpinit(xinit,@sincos)
```

```
sol = bvp4c(@exampl,@border,solinit)
```

Результат приведен табл. 14.4, в которую для сравнения включен столбец `sin(sol.x)`, содержащий аналитическое решение краевой задачи.

Таблица 14.4

sol.x	sol.y(1)	sol.y(2)	sol.yp(1)	sol.yp(2)	sin(sol.x)
0	0	1.0000	1.0000	0	0
0.1963	0.1951	0.9808	0.9808	-0.1951	0.1951
0.3927	0.3827	0.9239	0.9239	-0.3827	0.3827
0.7854	0.7071	0.7071	0.7071	-0.7071	0.7071
1.1781	0.9239	0.3827	0.3827	-0.9239	0.9239
1.3744	0.9808	0.1951	0.1951	-0.9808	0.9808
1.5708	1.0000	0.0000	0.0000	-1.0000	1.0000