ТОМСКИЙ
ПОЛИТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ

tpu.ru

Углубленный курс информатики

Лекция 11
Численные методы решения систем дифференциальных уравнений

29 апреля 2022 г.

Вячеслав Алексеевич Чузлов,
к.т.н., доцент ОХИ ИШПР

ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Содержание

Численные методы решения систем дифференциальных уравнений

Задачи, в которых необходимо решить систему из нескольких дифференциальных уравнений с несколькими искомыми функциями, очень распространены в предметной области химической технологии.
Будем рассматривать системы, в которых число неизвестных функций совпадает с числом уравнений, разрешенных относительно производных.
К примеру, система из двух уравнений с двумя неизвестными функциями y и z от одного и того же аргумента x имеет вид:

$$\begin{cases} y' = f_1(x, y, z) \\ z' = f_2(x, y, z) \end{cases} \tag{1}$$

при этом штрих означает производную по x.

Численные методы решения систем дифференциальных уравнений

Общий вид системы из n уравнений с n неизвестными функциями $x_1, x_2, \ldots, x_n$ от переменной t имеет вид:

$$\begin{cases} \dfrac{dx_1}{dt} = f_1\left(t, x_1, x_2, \ldots, x_n\right) \\ \dfrac{dx_2}{dt} = f_2\left(t, x_1, x_2, \ldots, x_n\right) \\ \quad \ldots \\ \dfrac{dx_n}{dt} = f_n\left(t, x_1, x_2, \ldots, x_n\right) \end{cases} \qquad (2)$$

Ранее мы рассмотрели численные методы решения обыкновенных дифференциальных уравнений вида $y' = f(x, y)$ (методы Эйлера и Рунге-Кутты). Данные методы применяются и в случае решения систем обыкновенных дифференциальных уравнений.

ТОМСКИЙ
ПОЛИТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ

Метод Эйлера

Метод Эйлера

Пусть дана следующая система обыкновенных дифференциальных уравнений:

$$\begin{cases} \dfrac{dy_1}{dx} = f_1\left(x, y_1, y_2\right) \\ \dfrac{dy_2}{dx} = f_2\left(x, y_1, y_2\right) \end{cases} \tag{3}$$

с начальными условиями:

$$\begin{gathered} y_1\big|_{x=x_0} = y_{01} \\ y_2\big|_{x=x_0} = y_{02} \end{gathered} \tag{4}$$

При использовании метода Эйлера, расчетные формулы примут следующий вид:

$$\begin{cases} y_{(i),1} = y_{i-1,1} + h \cdot f_1\left(x_{i-1}, y_{(i-1),1}, y_{(i-1),2}\right) \\ y_{(i),2} = y_{i-1,2} + h \cdot f_2\left(x_{i-1}, y_{(i-1),1}, y_{(i-1),2}\right) \\ \quad x_i = x_{i-1} + h \end{cases} \tag{5}$$

где h – шаг интегрирования; $f_1\left(x_i, y_{i,1}, y_{i,2}\right)$ и $f_2\left(x_i, y_{i,1}, y_{i,2}\right)$ – правые части дифференциальных уравнений.

Пример 1

Пусть требуется решить систему дифференциальных уравнений первого порядка:

$$\begin{cases} \dfrac{dy_1}{dx} = y_2 \\ \dfrac{dy_2}{dx} = e^{-x \cdot y_1} \end{cases}$$

методом Эйлера на отрезке $[0, 1]$ с шагом $h = 0.1$. Начальные условия: $x_0 = 0$; $y_1(0) = 0$; $y_2(0) = 0$.
Воспользуемся формулой (5) и запишем выражения для $y_{i,1}$ и $y_{i,2}$:

$$\begin{cases} y_{i,1} = y_{(i-1),1} + 0.1 \cdot y_{(i-1),2} \\ y_{i,2} = y_{(i-1),2} + 0.1 \cdot e^{-x_{i-1} \cdot y_{(i-1),1}} \\ \quad x_i = x_{i-1} + h \end{cases}$$

Пример 1

Результаты вычислений сведем в таблице.

i	x_i	$y_{i,1} = y_{(i-1),1} + 0.1 \cdot y_{(i-1),2}$	$y_{i,2} = y_{(i-1),2} + 0.1 \cdot e^{-x_{i-1} \cdot y_{(i-1),1}}$
0	0.0	0.0000	0.0000
1	0.1	0.0000	0.1000
2	0.2	0.0100	0.2000
3	0.3	0.0300	0.2998
4	0.4	0.0600	0.3989
5	0.5	0.0999	0.4965
6	0.6	0.1495	0.5917
7	0.7	0.2087	0.6831
8	0.8	0.2770	0.7695
9	0.9	0.3539	0.8496
10	1.0	0.4389	0.9223

Программная реализация

```python
import math


#Функция, содержащая правые части дифференциальных уравнений
def equations(x, y):
    return [y[1], math.exp(-x * y[0])]


def eiler(func, x0, xf, y0, h):
    count = int((xf - x0) / h) + 1
    y = [y0[:]] #создание массива y с начальными условиями
    x = x0

    for i in range(1, count):
        right_parts = func(x, y[i-1])
        y.append([]) #добавление пустой строки

        for j in range(len(y0)):
            y[i].append(y[i-1][j] + h * right_parts[j])

        x += h

    return y
```

```python
print(eiler(equations, 0, 1, [0, 0], 0.1))

```

```
[[0, 0],
 [0.0, 0.1],
 [0.010000000000000002, 0.2],
 [0.030000000000000006, 0.2998001998667334],
 [0.05998001998667334, 0.3989042377440217 3],
 [0.09987044376107551, 0.4965335889709902],
 [0.14952380265817453, 0.5916626935059732],
 [0.20869007200877185, 0.6830819284599525],
 [0.2769982648547671, 0.7694905222116074],
 [0.3539473170759278 6, 0.849614212888738 7],
 [0.4389087383648017, 0.922334296571365 7]]
```

Пример 2

Закон действующих масс

Скорость химической реакции прямо пропорциональна произведению концентраций реагирующих веществ, возведенных в степени, равные их стехиометрическим коэффициентам.

Схема химической реакции:

$$n_1 A_1 + n_2 A_2 + n_3 A_3 \xrightarrow{k} B$$

Скорость данной реакции:

$$r = k \cdot [A_1]^{n_1} \cdot [A_2]^{n_2} \cdot [A_3]^{n_3}$$

Изменение концентрации каждого компонента во времени:

$$\begin{cases} \dfrac{\partial C_{A_1}}{\partial t} = -n_1 \cdot r \\ \dfrac{\partial C_{A_2}}{\partial t} = -n_2 \cdot r \\ \dfrac{\partial C_{A_3}}{\partial t} = -n_3 \cdot r \\ \dfrac{\partial C_B}{\partial t} = r \end{cases}$$

где k – константа скорости химической реакции; C_{A_1}, C_{A_2}, C_{A_3}, C_B – концентрации веществ (моль/л), участвующих в химической реакции, n_1, n_2, n_3 – стехиометрические коэффициенты в уравнении реакции.

Пример 2

Пусть дана схема химических реакций:

$$A \underset{k_2}{\overset{k_1}{\rightleftarrows}} B$$

Скорость прямой реакции: $r_1 = k_1 \cdot C_A$; скорость обратной реакции: $r_2 = k_2 \cdot C_B$. Константы скоростей реакций: $k_1 = 0.85$; $k_2 = 0.1$, C_A и C_B – концентрации компонентов A и B. Изменение концентрации реагирующих веществ во времени описывается следующей системой дифференциальных уравнений:

$$\begin{cases} \dfrac{\partial C_A}{\partial t} = -r_1 + r_2 \\ \dfrac{\partial C_B}{\partial t} = r_1 - r_2 \end{cases}$$

Необходимо определить изменение концентрации каждого компонента по времени методом Эйлера на отрезке $[0, 1]$ с шагом $h = 0.1$. Начальные условия: $C_A(0) = 1$ (моль/л); $C_B(0) = 0$ (моль/л).
Воспользуемся формулой (5) и запишем выражения для $C_{A,i}$ и $C_{B,i}$:

$$\begin{cases} C_{A,i} = C_{A,(i-1)} + 0.1 \cdot \left(-k_1 \cdot C_{A,(i-1)} + k_2 \cdot C_{B,(i-1)}\right) \\ C_{B,i} = C_{B,(i-1)} + 0.1 \cdot \left(k_1 \cdot C_{A,(i-1)} - k_2 \cdot C_{B,(i-1)}\right) \\ \quad t_i = t_{i-1} + h \end{cases}$$

Пример 2

$$\begin{cases} C_{A,i} = C_{A,(i-1)} + 0.1 \cdot \left(-k_1 \cdot C_{A,(i-1)} + k_2 \cdot C_{B,(i-1)}\right) \\ C_{B,i} = C_{B,(i-1)} + 0.1 \cdot \left(k_1 \cdot C_{A,(i-1)} - k_2 \cdot C_{B,(i-1)}\right) \\ \quad t_i = t_{i-1} + h \end{cases}$$

Результаты вычислений сведем в таблице.

i	t_i	$C_{A,i}$	$C_{B,i}$
0	0.0	1.0000	0.0000
1	0.1	0.9150	0.0850
2	0.2	0.8381	0.1619
3	0.3	0.7685	0.2315
4	0.4	0.7055	0.2945
5	0.5	0.6484	0.3516
6	0.6	0.5968	0.4032
7	0.7	0.5501	0.4499
8	0.8	0.5079	0.4921
9	0.9	0.4696	0.5304
10	1.0	0.4350	0.5650

Программная реализация


```python
def equations(t, c, k):  #Функция, содержащая правые части дифф. уравнений
    right_parts = [
        -k[0] * c[0] + k[1] * c[1],
        k[0] * c[0] - k[1] * c[1],
    ]
    return right_parts


def eiler(func, x0, xf, y0, h, args=()):
    count = int((xf - x0) / h) + 1
    y = [y0[:]]
    x = x0

    for i in range(1, count):
        right_parts = func(x, y[i-1], *args)
        y.append([])

        for j in range(len(y0)):
            y[i].append(y[i-1][j] + h * right_parts[j])

        x += h

    return y
```

Программная реализация

```
k = [0.85, 0.1]
print(eiler(equations, 0, 1, [1, 0], 0.1, args=(k, )))

```

```
[[1, 0],
 [0.915, 0.085],
 [0.838075, 0.16192500000000004],
 [0.7684578750000001, 0.23154212500000004],
 [0.7054543768750001, 0.29454562312500004],
 [0.6484362110718751, 0.3515637889281250б],
 [0.596834771020047, 0.4031652289799532],
 [0.5501354677731425, 0.4498645322268577],
 [0.5078725983346939, 0.4921274016653062],
 [0.469624701492898, 0.5303752985071022],
 [0.4350103548510727, 0.5649896451489275]]
```

- В реализацию функции `eiler` был добавлен опциональный параметр `args`, который по умолчанию равен пустому кортежу `()`, необходимый в тех случаях, когда функция правых частей дифференциальных уравнений помимо параметров `x` и `y` принимает еще дополнительные аргументы, именно их нужно будет передать в кортеже `args` при вызове функции `eiler`, используя механизм передачи произвольного количества аргументов функции (оператор `*`).
- В нашем случае в качестве дополнительного аргумента нужно передать список констант скоростей реакций через кортеж `args`.

ТОМСКИЙ
ПОЛИТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ

Метод Рунге-Кутты

Метод Рунге-Кутты

Пусть дана следующая система обыкновенных дифференциальных уравнений:

$$\begin{cases} \dfrac{dy_1}{dx} = f_1\left(x, y_1, y_2\right) \\ \dfrac{dy_2}{dx} = f_2\left(x, y_1, y_2\right) \end{cases} \tag{6}$$

с начальными условиями:

$$\begin{aligned} y_1\big|_{x=x_0} &= y_{01} \\ y_2\big|_{x=x_0} &= y_{02} \end{aligned} \tag{7}$$

Метод Рунге-Кутты

При использовании метода Рунге-Кутты, расчетные формулы примут следующий вид:

$$\begin{cases} y_{i,1} = y_{(i-1),1} + h/6 \cdot (k_{1,1} + 2 \cdot k_{2,1} + 2 \cdot k_{3,1} + k_{4,1}) \\ y_{i,2} = y_{(i-1),2} + h/6 \cdot (k_{1,2} + 2 \cdot k_{2,2} + 2 \cdot k_{3,2} + k_{4,2}) \\ x_i = x_{i-1} + h \end{cases} \tag{8}$$

где

$$\begin{aligned}
k_{1,1} &= f_1\left(x, y_{(i-1),1}, y_{(i-1),2}\right); & k_{1,2} &= f_2\left(x, y_{(i-1),1}, y_{(i-1),2}\right); \\
k_{2,1} &= f_1\left(x + \frac{h}{2}, y_{(i-1),1} + k_{1,1} \cdot \frac{h}{2}, y_{(i-1),2} + k_{1,2} \cdot \frac{h}{2}\right); & k_{2,2} &= f_2\left(x + \frac{h}{2}, y_{(i-1),1} + k_{1,1} \cdot \frac{h}{2}, y_{(i-1),2} + k_{1,2} \cdot \frac{h}{2}\right); \\
k_{3,1} &= f_1\left(x + \frac{h}{2}, y_{(i-1),1} + k_{2,1} \cdot \frac{h}{2}, y_{(i-1),2} + k_{2,2} \cdot \frac{h}{2}\right); & k_{3,2} &= f_2\left(x + \frac{h}{2}, y_{(i-1),1} + k_{2,1} \cdot \frac{h}{2}, y_{(i-1),2} + k_{2,2} \cdot \frac{h}{2}\right); \\
k_{4,1} &= f_1\left(x + h, y_{(i-1),1} + k_{3,1} \cdot h, y_{(i-1),2} + k_{3,2} \cdot h\right); & k_{4,2} &= f_2\left(x + h, y_{(i-1),1} + k_{3,1} \cdot h, y_{(i-1),2} + k_{3,2} \cdot h\right).
\end{aligned} \tag{9}$$

где h – шаг интегрирования; $f_1\left(x_i, y_{(i-1),1}, y_{(i-1),2}\right)$ и $f_2\left(x_i, y_{(i-1),1}, y_{(i-1),2}\right)$ – правые части дифференциальных уравнений, $k_{1,j}$, $k_{2,j}$, $k_{3,j}$, $k_{4,j}$ – параметры метода Рунге-Кутты для j-го уравнения.

Пример 1

Решим методом Рунге-Кутты пример, приведенный на слайде 7. Воспользуемся формулами (8), (9) и запишем выражения для нахождения значений искомых переменных $y_{i,1}$ и $y_{i,2}$:

$$
\begin{aligned}
k_{1,1} &= y_{(i-1),2}; & k_{1,2} &= \exp\left(-x_i \cdot y_{(i-1),1}\right); \\
k_{2,1} &= y_{(i-1),2} + k_{1,2} \cdot \frac{h}{2}; & k_{2,2} &= \exp\left[-\left(x_i + \frac{h}{2}\right) \cdot \left(y_{(i-1),1} + k_{1,1} \cdot \frac{h}{2}\right)\right] \\
k_{3,1} &= y_{(i-1),2} + k_{2,2} \cdot \frac{h}{2}; & k_{3,2} &= \exp\left[-\left(x_i + \frac{h}{2}\right) \cdot \left(y_{(i-1),1} + k_{2,1} \cdot \frac{h}{2}\right)\right] \\
k_{4,1} &= y_{(i-1),2} + k_{3,2} \cdot h; & k_{4,2} &= \exp\left[-\left(x_i + h\right) \cdot \left(y_{(i-1),1} + k_{3,1} \cdot h\right)\right]
\end{aligned}
$$

$$
\begin{cases}
y_{i,1} = y_{(i-1),1} + \dfrac{0.1}{6} \cdot (k_{1,1} + 2 \cdot k_{2,1} + 2 \cdot k_{3,1} + k_{4,1}) \\
y_{i,2} = y_{(i-1),2} + \dfrac{0.1}{6} \cdot (k_{1,2} + 2 \cdot k_{2,2} + 2 \cdot k_{3,2} + k_{4,2}) \\
\quad x_i = x_{i-1} + 0.1
\end{cases}
$$

Пример 1

Результаты вычислений сведем в таблице.

i	x_i	$k_{1,1}$	$k_{2,1}$	$k_{3,1}$	$k_{4,1}$	$y_{i,1}$	$k_{1,2}$	$k_{2,2}$	$k_{3,2}$	$k_{4,2}$	$y_{i,2}$
0	0.0	—	—	—	—	0.0000	—	—	—	—	0.0000
1	0.1	0.0000	0.0500	0.0500	0.1000	0.0050	1.0000	1.0000	0.9999	0.9995	0.1000
2	0.2	0.1000	0.1500	0.1499	0.1998	0.0200	0.9995	0.9985	0.9981	0.9960	0.1998
3	0.3	0.1998	0.2496	0.2494	0.2990	0.0449	0.9960	0.9925	0.9919	0.9866	0.2990
4	0.4	0.2990	0.3483	0.3480	0.3968	0.0797	0.9866	0.9793	0.9784	0.9686	0.3968
5	0.5	0.3968	0.4453	0.4446	0.4923	0.1242	0.9686	0.9562	0.9551	0.9398	0.4924
6	0.6	0.4924	0.5393	0.5384	0.5844	0.1781	0.9398	0.9214	0.9202	0.8987	0.5844
7	0.7	0.5844	0.6293	0.6281	0.6716	0.2409	0.8987	0.8739	0.8727	0.8448	0.6717
8	0.8	0.6717	0.7139	0.7124	0.7529	0.3122	0.8448	0.8139	0.8126	0.7790	0.7529
9	0.9	0.7529	0.7919	0.7901	0.8271	0.3913	0.7790	0.7427	0.7415	0.7032	0.8271
10	1.0	0.8271	0.8623	0.8603	0.8933	0.4774	0.7032	0.6630	0.6619	0.6204	0.8933

Программная реализация


```python
import math

def equations(x, y): #Функция, содержащая правые части дифф. уравнений
    return [y[1], math.exp(-x * y[0])]

def rk(func, x0, xf, y0, h):
    count = int((xf - x0) / h) + 1
    y = [y0[:]]
    x = x0

    for i in range(1, count):
        k1 = func(x, y[i-1])
        k2 = func(x + h / 2, list(map(
            lambda arr1, arr2: arr1 + arr2 * h / 2, y[i-1], k1)))
        k3 = func(x + h / 2, list(map(
            lambda arr1, arr2: arr1 + arr2 * h / 2, y[i-1], k2)))
        k4 = func(x + h, list(map(
            lambda arr1, arr2: arr1 + arr2 * h, y[i-1], k3)))

        y.append([])

        for j in range(len(y0)):
            y[i].append(y[i-1][j] + h / 6 * (k1[j] + 2 * k2[j] + 2 * k3[j] + k4[j]))

        x += h

    return y
```

Программная реализация

```python
print(rk(equations, 0, 1, [0, 0], 0.1))
```

```
[[0, 0],
 [0.004999791679686959, 0.09998750234339197],
 [0.019992089353337197, 0.1998002782423727 3],
 [0.04493954532954178, 0.2989921821997826],
 [0.07974589273138522, 0.3968347761839209 3],
 [0.12422922613072 27, 0.4923515428080233 5],
 [0.1781000081292174, 0.5843789596377397],
 [0.2409466243210469 6, 0.67165612248553],
 [0.3122311354618596, 0.752937520153815 3],
 [0.39129695254854, 0.827116006499604 7],
 [0.4773885589403407, 0.893337443498574 7]]
```

Программная реализация

Альтернативный вариант реализации метода Рунге-Кутты:

```python
def rk(func, x0, xf, y0, h):
    count = int((xf - x0) / h) + 1
    y = [y0[:]]
    x = x0

    for i in range(1, count):
        k1 = func(x, y[i-1])
        k2 = func(x + h / 2, [y + k * h / 2 for y, k in zip(y[i-1], k1)])
        k3 = func(x + h / 2, [y + k * h / 2 for y, k in zip(y[i-1], k2)])
        k4 = func(x + h, [y + k * h for y, k in zip(y[i-1], k3)])

        y.append([])

        for j in range(len(y0)):
            y[i].append(
                y[i-1][j] + h / 6 * (k1[j] + 2 * k2[j] + 2 * k3[j] + k4[j])
            )

        x += h

    return y
```

Пример 2

Рассмотрим также решение примера, приведенного на слайде 12, методом Рунге-Кутты. Воспользуемся формулами (8), (9) и запишем выражения для нахождения значений искомых концентраций компонентов $C_{A,i}$ и $C_{B,i}$:

$$\begin{cases} C_{A,i} = C_{A,(i-1)} + \dfrac{0.1}{6} \cdot (k_{1,1} + 2 \cdot k_{2,1} + 2 \cdot k_{3,1} + k_{4,1}) \\ C_{B,i} = C_{B,(i-1)} + \dfrac{0.1}{6} \cdot (k_{1,2} + 2 \cdot k_{2,2} + 2 \cdot k_{3,2} + k_{4,2}) \\ t_i = t_{i-1} + 0.1 \end{cases}$$

$k_{\mathbf{1},\mathbf{1}} = -k_{\mathbf{1}} \cdot C_{A,(i-\mathbf{1})} + k_{\mathbf{2}} \cdot C_{B,(i-\mathbf{1})}$;

$k_{\mathbf{2},\mathbf{1}} = -k_{\mathbf{1}} \cdot \left(C_{A,(i-\mathbf{1})} + k_{\mathbf{1},\mathbf{1}} \cdot \dfrac{h}{2}\right) + k_{\mathbf{2}} \cdot \left(C_{B,(i-\mathbf{1})} + k_{\mathbf{1},\mathbf{2}} \cdot \dfrac{h}{2}\right)$;

$k_{\mathbf{3},\mathbf{1}} = -k_{\mathbf{1}} \cdot \left(C_{A,(i-\mathbf{1})} + k_{\mathbf{2},\mathbf{1}} \cdot \dfrac{h}{2}\right) + k_{\mathbf{2}} \cdot \left(C_{B,(i-\mathbf{1})} + k_{\mathbf{2},\mathbf{2}} \cdot \dfrac{h}{2}\right)$;

$k_{\mathbf{4},\mathbf{1}} = -k_{\mathbf{1}} \cdot \left(C_{A,(i-\mathbf{1})} + k_{\mathbf{3},\mathbf{1}} \cdot h\right) + k_{\mathbf{2}} \cdot \left(C_{B,(i-\mathbf{1})} + k_{\mathbf{3},\mathbf{2}} \cdot h\right)$;

$k_{\mathbf{1},\mathbf{2}} = k_{\mathbf{1}} \cdot C_{A,(i-\mathbf{1})} - k_{\mathbf{2}} \cdot C_{B,(i-\mathbf{1})}$;

$k_{\mathbf{2},\mathbf{2}} = k_{\mathbf{1}} \cdot \left(C_{A,(i-\mathbf{1})} + k_{\mathbf{1},\mathbf{1}} \cdot \dfrac{h}{2}\right) - k_{\mathbf{2}} \cdot \left(C_{B,(i-\mathbf{1})} + k_{\mathbf{1},\mathbf{2}} \cdot \dfrac{h}{2}\right)$;

$k_{\mathbf{3},\mathbf{2}} = k_{\mathbf{1}} \cdot \left(C_{A,(i-\mathbf{1})} + k_{\mathbf{2},\mathbf{1}} \cdot \dfrac{h}{2}\right) - k_{\mathbf{2}} \cdot \left(C_{B,(i-\mathbf{1})} + k_{\mathbf{2},\mathbf{2}} \cdot \dfrac{h}{2}\right)$;

$k_{\mathbf{4},\mathbf{2}} = k_{\mathbf{1}} \cdot \left(C_{A,(i-\mathbf{1})} + k_{\mathbf{3},\mathbf{1}} \cdot h\right) - k_{\mathbf{2}} \cdot \left(C_{B,(i-\mathbf{1})} + k_{\mathbf{3},\mathbf{2}} \cdot h\right)$

Пример 2

Результаты вычислений сведем в таблице.

i	t_i	$k_{\mathbf{1},\mathbf{1}}$	$k_{\mathbf{2},\mathbf{1}}$	$k_{\mathbf{3},\mathbf{1}}$	$k_{\mathbf{4},\mathbf{1}}$	$C_{A,i}$	$k_{\mathbf{1},\mathbf{2}}$	$k_{\mathbf{2},\mathbf{2}}$	$k_{\mathbf{3},\mathbf{2}}$	$k_{\mathbf{4},\mathbf{2}}$	$C_{B,i}$
0	0.0	—	—	—	—	1.0000	—	—	—	—	0.0000
1	0.1	−0.8500	−0.8096	−0.8115	−0.7729	0.9189	0.8500	0.8096	0.8115	0.7729	0.0811
2	0.2	−0.7730	−0.7363	−0.7380	−0.7029	0.8452	0.7730	0.7363	0.7380	0.7029	0.1548
3	0.3	−0.7029	−0.6695	−0.6711	−0.6392	0.7781	0.7029	0.6695	0.6711	0.6392	0.2219
4	0.4	−0.6392	−0.6088	−0.6103	−0.5812	0.7171	0.6392	0.6088	0.6103	0.5812	0.2829
5	0.5	−0.5813	−0.5537	−0.5550	−0.5286	0.6617	0.5813	0.5537	0.5550	0.5286	0.3383
6	0.6	−0.5286	−0.5035	−0.5047	−0.4807	0.6113	0.5286	0.5035	0.5047	0.4807	0.3887
7	0.7	−0.4807	−0.4579	−0.4589	−0.4371	0.5654	0.4807	0.4579	0.4589	0.4371	0.4346
8	0.8	−0.4371	−0.4164	−0.4174	−0.3975	0.5237	0.4371	0.4164	0.4174	0.3975	0.4763
9	0.9	−0.3975	−0.3786	−0.3795	−0.3615	0.4858	0.3975	0.3786	0.3795	0.3615	0.5142
10	1.0	−0.3615	−0.3443	−0.3451	−0.3287	0.4513	0.3615	0.3443	0.3451	0.3287	0.5487

Программная реализация


```
def equations(t, c, k):  #Функция, содержащая правые части дифф. уравнений
    right_parts = [-k[0] * c[0] + k[1] * c[1],
                    k[0] * c[0] - k[1] * c[1]]
    return right_parts

def rk(func, x0, xf, y0, h, args=()):
    count = int((xf - x0) / h) + 1
    y = [y0[:]]
    x = x0

    for i in range(1, count):
        k1 = func(x, y[i-1], *args)
        k2 = func(x + h / 2, list(map(
            lambda arr1, arr2: arr1 + arr2 * h / 2, y[i-1], k1)), *args)
        k3 = func(x + h / 2, list(map(
            lambda arr1, arr2: arr1 + arr2 * h / 2, y[i-1], k2)), *args)
        k4 = func(x + h, list(map(
            lambda arr1, arr2: arr1 + arr2 * h, y[i-1], k3)), *args)
        y.append([])

        for j in range(len(y0)):
            y[i].append(y[i-1][j] + h / 6 * (k1[j] + 2 * k2[j] + 2 * k3[j] + k4[j]))

        x += h

    return y
```

Программная реализация

```python
k = [0.85, 0.1]
print(rk(equations, 0, 1, [1, 0], 0.1, args=(k, )))
```

```
[[1, 0],
 [0.9189126823697916, 0.0810873176302084],
 [0.8451740652412765, 0.1548259347587235],
 [0.7781181579189691, 0.2218818420810309],
 [0.71713932644751 4, 0.28286067355248 6],
 [0.6616868236612737, 0.3383131763387262 6],
 [0.6112598149591241, 0.388740185040876],
 [0.5654028548783672, 0.4345971451216329],
 [0.5237017736131901, 0.4762982263868099],
 [0.4857799363256236, 0.5142200636743764],
 [0.4512948414639336, 0.5487051585360664]]
```

Программная реализация

Альтернативный вариант реализации метода Рунге-Кутты:

```python
def rk(func, x0, xf, y0, h, args=()):
    count = int((xf - x0) / h) + 1
    y = [y0[:]]
    x = x0

    for i in range(1, count):
        k1 = func(x, y[i-1], *args)
        k2 = func(x + h / 2, [y + k * h / 2 for y, k in zip(y[i-1], k1)], *args)
        k3 = func(x + h / 2, [y + k * h / 2 for y, k in zip(y[i-1], k2)], *args)
        k4 = func(x + h, [y + k * h for y, k in zip(y[i-1], k3)], *args)

        y.append([])

        for j in range(len(y0)):
            y[i].append(y[i-1][j] + h / 6 * (k1[j] + 2 * k2[j] + 2 * k3[j] + k4[j]))

        x += h

    return y
```

Программная реализация

Для более аккуратного вывода результатов вычислений, код вызова функции `rk` можно переписать в следующем виде:

```python
k = [0.85, 0.1]
results = rk(equations, 0, 1, [1, 0], 0.1, args=(k, ))
for i, line in enumerate(results):
    ca, cb = line
    print(f"CA,{i:2} = {ca:.4f};  CB,{i:2} = {cb:.4f}")

```

```
CA, 0 = 1.0000;  CB, 0 = 0.0000
CA, 1 = 0.9189;  CB, 1 = 0.0811
CA, 2 = 0.8452;  CB, 2 = 0.1548
CA, 3 = 0.7781;  CB, 3 = 0.2219
CA, 4 = 0.7171;  CB, 4 = 0.2829
CA, 5 = 0.6617;  CB, 5 = 0.3383
CA, 6 = 0.6113;  CB, 6 = 0.3887
CA, 7 = 0.5654;  CB, 7 = 0.4346
CA, 8 = 0.5237;  CB, 8 = 0.4763
CA, 9 = 0.4858;  CB, 9 = 0.5142
CA,10 = 0.4513;  CB,10 = 0.5487
```

Программная реализация

Альтернативный вариант:

```
k = [0.85, 0.1]
results = rk(equations, 0, 1, [1, 0], 0.1, args=(k, ))
print("  CA    |    CB")
print("-" * 15)
for line in results:
    ca, cb = line
    print(f"{ca:.4f} | {cb:.4f}")

```

```
  CA    |    CB
---------------
1.0000 | 0.0000
0.9189 | 0.0811
0.8452 | 0.1548
0.7781 | 0.2219
0.7171 | 0.2829
0.6617 | 0.3383
0.6113 | 0.3887
0.5654 | 0.4346
0.5237 | 0.4763
0.4858 | 0.5142
0.4513 | 0.5487
```

Графическая визуализация

Построим графическую визуализацию полученного решения:

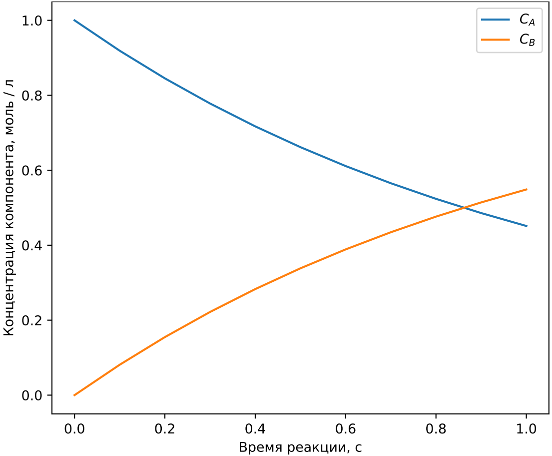

Рисунок 1 – Изменение концентрации реагирующих веществ во времени

Расчет схемы химических реакций

Рассмотрим следующую схему химических реакций:

$$A \to B \to C$$

с константами скоростей k_1 и k_2. Уравнения, описывающие скорость изменения концентраций компонентов по времени, записываются следующим образом:

$$\begin{cases} \dfrac{d\,[A]}{dt} = -k_1 \cdot [A] \\ \dfrac{d\,[B]}{dt} = k_1 \cdot [A] - k_2 \cdot [B] \\ \dfrac{d\,[C]}{dt} = k_2 \cdot [B] \end{cases}$$

Расчет схемы химических реакций

В данном случае можно решить эту систему обыкновенных дифференциальных уравнений аналитически, однако для численного решения предположим $y_1 \equiv [A]$, $y_2 \equiv [B]$ и $y_3 \equiv [C]$:

$$\begin{cases} \dfrac{d\,[A]}{dt} = -k_1 \cdot y_1 \\ \dfrac{d\,[B]}{dt} = k_1 \cdot y_1 - k_2 \cdot y_2 \\ \dfrac{d\,[C]}{dt} = k_2 \cdot y_2 \end{cases}$$

Зададимся значениями констант: $k_1 = 0.2\ c^{-1}$, $k_2 = 0.8\ c^{-1}$ и начальными условиями: $y_1(0) = 100$, $y_2(0) = 0\ y_3(0) = 0$.

Метод Эйлера


```
def func(t, y, k):
    return [-k[0] * y[0],
            k[0] * y[0] - k[1] * y[1],
            k[1] * y[1]]

def eiler(func, x0, xf, y0, h, args=()):
    count = int((xf - x0) / h) + 1
    y = [y0[:]]
    x = x0

    for i in range(1, count):
        right_parts = func(x, y[i-1], *args)
        y.append([])

        for j in range(len(y0)):
            y[i].append(y[i-1][j] + h * right_parts[j])

        x += h

    return y

k, y0 = [0.2, 0.8], [100, 0, 0]
t0, tf = 0, 20
y_eiler = eiler(func, t0, tf, y0, 0.1, args=(k, ))
```

Метод Рунге-Кутты

```
def func(t, y, k):
    return [-k[0] * y[0],
            k[0] * y[0] - k[1] * y[1],
            k[1] * y[1]]

def rk(func, x0, xf, y0, h, args=()):
    count = int((xf - x0) / h) + 1
    x, y = x0, [y0[:]]

    for i in range(1, count):
        k1 = func(x, y[i-1], *args)
        k2 = func(x + h / 2, [y + k * h / 2 for y, k in zip(y[i-1], k1)], *args)
        k3 = func(x + h / 2, [y + k * h / 2 for y, k in zip(y[i-1], k2)], *args)
        k4 = func(x + h, [y + k * h for y, k in zip(y[i-1], k3)], *args)
        y.append([])

        for j in range(len(y0)):
            y[i].append(y[i-1][j] + h / 6 * (k1[j] + 2 * k2[j] + 2 * k3[j] + k4[j]))

        x += h

    return y

k, y0 = [0.2, 0.8], [100, 0, 0]
t0, tf = 0, 20
y_rk = rk(func, t0, tf, y0, 0.1, args=(k, ))
```

Графическая визуализация

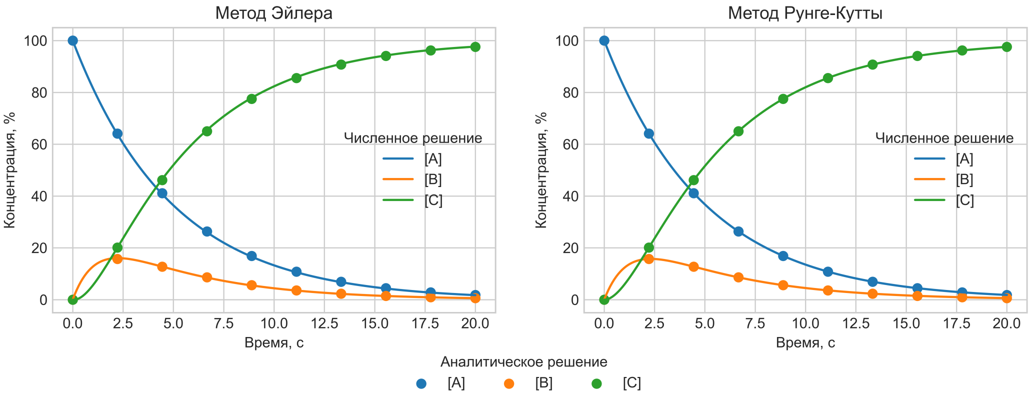

Рисунок 2 – Изменение концентрации реагирующих веществ во времени

ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Контакты

Вячеслав Алексеевич Чузлов,
к.т.н., доцент ОХИ ИШПР

Учебный корпус №2, ауд. 136
chuva@tpu.ru
+7-962-782-66-15

Благодарю за внимание!