РЕШЕНИЕ ОБЫКНОВЕННЫХ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ

Многие физические явления могут быть описаны обыкновенными дифференциальными уравнениями (ОДУ), т.е. такими уравнениями, которые содержат производные только по одной единственной координате.

ОДУ в электротехнике:

- переходные процессы в сети.
- процесс гашения электрической дуги (каналовые модели)
- взаимодействие электрической дуги и сети(модели черного ящика)
- Задачи оптимизации (нахождение наилучших конструктивных решений при заданном числе ограничений)

ОБЫКНОВЕННЫЕ ДИФФЕРЕНЦИАЛЬНЫЕ УРАВНЕНИЯ

Начальная задача (задача Коши)

Краевая задача

Задача Коши — одна из основных задач теории дифференциальных уравнений; состоит в нахождении решения (интеграла) дифференциального уравнения (системы дифференциальных уравнений), удовлетворяющего начальным условиям (начальным данным).

В задаче Коши область, в которой должно быть определено искомое решение, **заранее не указывается**.

Модельная задача.

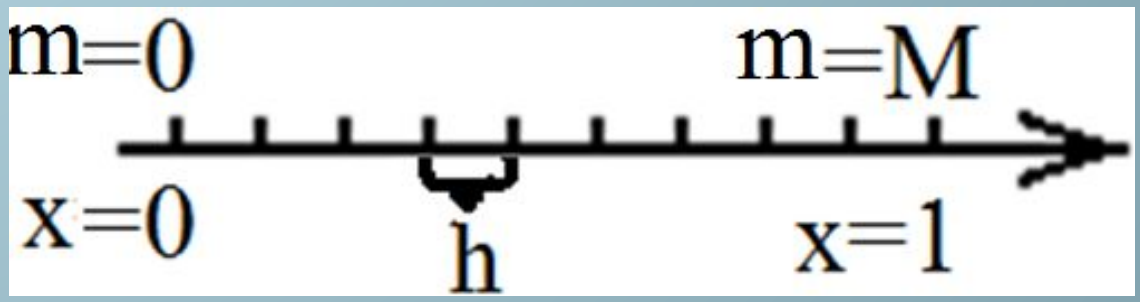
$$\begin{cases} \dfrac{dU(x)}{dx} = -A \cdot U(x) \\ x \in (0,1] \\ U\big|_{x=0} = U_o \end{cases} \qquad (1)$$

$$U(x) = U_o \cdot e^{-A \cdot x}$$

Дискретное пространство – это совокупность узлов сетки, характеризуемых шагом (const или нет) и означаемым h-параметром

Модель:

разобьём ось x на промежутке (0,1] на M равных частей и получим дискретное пространство



$$h = \frac{1}{M} \qquad x_m = 0 + m \cdot h, \quad m = 0 \ldots M$$

$$\frac{dU}{dx} = \lim_{\Delta x \to 0} \frac{\Delta U}{\Delta x}$$

$$\frac{U_{m+1} - U_m}{h} = -A \cdot U_m, \qquad m=\overline{0,M-1}$$

Записанная в таком виде задача представляет собой <u>аппроксимацию</u> дифференциального уравнения на дискретном пространстве h (на сетке h).

$$U_{m+1} = (1 - A \cdot h) \cdot U_m$$

$m=\overline{0,M-1}$

Если в правой части стоит функция общего вида $f(x,y(x))$, то наша схема запишется таким образом:

$$y_k = y_{k-1} + f\left(x_{k-1}, y_{k-1}\right)h$$

Это **<u>схема Эйлера</u>**

Разностная формула представляет собой **разностную схему** для модельной дифференциальной задачи на дискретном пространстве h (или просто на сетке).

Конфигурацию узлов, используемую для разностной записи уравнений на сетке, называют **шаблоном.**

Шаблон, используемый на предыдущем слайде для аппроксимации – двухточечный.

Поскольку начальное значение функции U известно, это U_0, то решение в любой точке нашего дискретного пространства находится по формуле: $U_{m+1} = (1 - A \cdot h) \cdot U_m$

$$U_{m+1} = (1 - A \cdot h)^{m+1} \cdot U_o \qquad m=\overline{0,M-1}$$

Значение сеточной функции в последней точке промежутка:

$$U_M = U_o \cdot (1 - \frac{A}{M})^M \underset{M \to 0 \Leftrightarrow h \to 0}{\approx} A \cdot e^{-A}$$

Значение сеточной функции в произвольной точке промежутка:

$$U_m = U_o \cdot (1 - A \cdot h)^m = U_o \cdot (1 - A \cdot h)^{\frac{xm}{h}} = U_o \cdot (1 - \frac{A \cdot x_m}{m})^m \underset{h \to 0}{\approx} U_o \cdot e^{-A \cdot x_m}$$

Если значение функции в точке, полученное в результате решения разностного уравнения стремиться, при уменьшении шага, к значению в этой точке функции, полученной в результате решения исходного дифференциального уравнения, то говорят, что имеется

сходимость.

Погрешность использованной разностной схемы будет **порядка h**.

Или иначе: данная **разностная схема имеет первый порядок точности**.

$$U(h) = U_o + \left(\frac{\partial U}{\partial x}\right)_0 \cdot h + \frac{1}{2} \cdot \left(\frac{\partial^2 U}{\partial x^2}\right)_0 \cdot h^2 + \frac{h^3}{6} \cdot \frac{\partial^3 U}{\partial x^3} + \ldots$$

$$U(x_m + h) = U(x_m) + h \cdot U'(x_m) + \frac{h^2}{2} \cdot U''(x_m) + \frac{h^3}{6} \cdot U'''(x_m) + \ldots$$

$$\frac{U(x_m + h) - U(x_m)}{h} = U'(x_m) + 0(h)$$

$$U'(x_m) = -A \cdot U_m$$

$$\frac{U(x_m + h) - U(x_m)}{h} = -A \cdot U_m + 0(h)$$

$$\frac{dU}{dx} = \lim_{\Delta x \to 0} \frac{\Delta U}{\Delta x}$$

$$\frac{U_{m+1} - U_m}{h} = -A \cdot U_m, \qquad m=\overline{0,M-1}$$

Записанная в таком виде задача представляет собой аппроксимацию дифференциального уравнения на на дискретном пространстве h (на сетке h).

Типы разностей!!!!!

Центральные разности

$$U(x_m - h) = U(x_m) - h \cdot U'(x_m) + \frac{h^2}{2} \cdot U''(x_m) - \frac{h^3}{6} \cdot U'''(x_m) + \ldots$$

$$U_{m+1} - U_{m-1} = 2h \cdot U'(x_m) + \frac{2}{6} h^3 \cdot U'''(x_m) + 0(h^4)$$

$$\frac{U_{m+1} - U_{m-1}}{2h} = U'(x_m) + \frac{1}{6} h^2 \cdot U'''(x_m) + 0(h^3)$$

$$\frac{U_{m+1} - U_{m-1}}{2h} = U'(x_m) + 0(h^2)$$

$$\frac{U_{m+1} - U_{m-1}}{2h} = -A \cdot U_m, \qquad m = \overline{0, M-1}$$

Формальные определения.

Введем следующие обозначения:

LU=f – дифференциальная задача (1)

U(x) - искомое решение (пусть непрерывное)

L - заданный дифференциальный оператор

f - заданная функция (правая часть)

$\omega^{(h)}$ - совокупность узлов разностной сетки, принадлежащая дискретному пространству D (считаем, что по любому направлению шаг постоянен).

h - шаг сетки

$U^{(h)}$ - сеточная функция

$L_h U^{(h)}=f^{(h)}$, L_h – заданный разностный оператор (2)

Сравнению функций U(x) и $U^{(h)}$ препятствует их разная функциональная природа. Снесём U(x) на сетку:

$$\left[U\right]_h\Big|_{x_m} = U(x_m)$$

проектирование решения на сетку

Сходимость: решение разностной задачи (2) $U^{(h)}$ сходится к решению задачи (1) если

$$\left\| U^h - [U]_h \right\|_{U^h} \xrightarrow[h \to 0]{} 0$$

в пространстве сеточных функций.

Если ,

$$\left\| U^h - [U]_h \right\|_{U^h} \le c \cdot h^k, \; c, k > 0$$

то имеет место сходимость порядка “k”.

Аппроксимация: разностная задача (2) аппроксимирует (1) на решение U(x), если

$$\left\| \delta \cdot f^{h} \right\|_{F_h} \xrightarrow[h \to 0]{} 0$$

где F_h - пространство сеточных функций правых частей, т.е.

$$L_h \left[U \right]_h = f^{(h)} + \delta f^{(h)}$$

и $\delta f^{(h)}$ - погрешность аппроксимации или невязка или если существуют такие C_1, k_1, большие нуля, что

$$\delta f^{(h)} \le C_1 h^{k_1}$$

то имеет место аппроксимация порядка h^{k_1}

Устойчивость:

$\exists \varepsilon^{n} \in F_n = f \quad \exists + \quad {}_h{}^h \quad z^{(h)} \quad \varepsilon^{(h)} \rightarrow {}^{(h)}$

если $\forall$ (не большого) $\varepsilon^{(h)}$ $\exists! z^{(h)}$ и $\left\| z^h - U^h \right\|_{U_n} \le C_2 \left\| \varepsilon^{(h)} \right\|_{F_n}$

Разностная схема устойчива, if $\exists$ такие два числа h_0 и $\delta > 0$:

$\forall$ для которых $\left\| \varepsilon^{(h)} \right\| \le \delta$ разностная схема $L_h z^h = f^h + \varepsilon^h$

имеет $! z^h$, удовлетворяющие неравенству $\left\| z^h - U^h \right\|_{U_h} \le C_2 \left\| \varepsilon^h \right\|_{F_h}$

$\forall\ h < h_0$

Другая формулировка: - разностная задача устойчива, если

$\exists h_0 > 0 : \forall h < h_0$ решение $L_h U^h = f^h$ является ! и ограниченным по норме: $\left\| U^h \right\| \le C_3 \left\| f^h \right\|$

Теорема Лакса: из аппроксимации и устойчивости следует сходимость.

Метод Рунге-Кутты

Семейство схем Рунге-Кутты основано на различной аппроксимации неизвестных аргументов y (t_n) в правых частях дифференциальных уравнений f(t,y).

Классический метод Рунге-Кутты используют для расчета стандартных моделей достаточно часто, так как при небольшом объеме вычислений он обладает точностью метода $O^4(h)$.

Данный метод не применим для жестких ОДУ.

Схемы Рунге-Кутты IV порядка точности

$$\begin{cases} K_1 = f(x_k,\ y_k) \\ K_2 = f\left(x_k + \frac{h}{2},\ y_k + \frac{h}{2} K_1\right) \\ K_3 = f\left(x_k + \frac{h}{2},\ y_k + \frac{h}{2} K_2\right) \\ K_4 = f(x_k + h,\ y_k + h K_3) \\ y_{k+1} = y_k + \frac{h}{6} \cdot (K_1 + 2 K_2 + 2 K_3 + K_4) \end{cases}$$

$$\begin{cases} K_1 = f(x_k,\ y_k) \\ K_2 = f\left(x_k + \frac{h}{4},\ y_k + \frac{h}{4} K_1\right) \\ K_3 = f\left(x_k + \frac{h}{2},\ y_k + \frac{h}{2} K_2\right) \\ K_4 = f(x_k + h,\ y_k + h K_1 - 2 h K_2 + 2 h K_3) \\ y_{k+1} = y_k + \frac{h}{6} \cdot (K_1 + 4 K_3 + K_4) \end{cases}$$

Метод Адамса-Бэшфорта

Метод основан на аппроксимации интерполяционными полиномами правых частей ОДУ. В зависимости от типа экстраполяции будут получаться алгоритмы различного порядка аппроксимации.

Методы Адамса-Бэшфорта могут быть как явными так и неявными.

Данный метод не применим для жестких ОДУ.

Схема Адамса-Бэшфорда

При решении ОДУ на *n*-м шаге численного решения имеем точную формулу интегрирования:

$$y_{n+1} = y_n + \int_{t_0}^{t_{n+1}} f(t, y(t))dt$$

Для построения схемы по значениям функции, вычисленным в *n* предшествующих узлах, строится интерполяционный полином – $L_{n-1}(x)$, который используется при интегрировании дифференциального уравнения. Интеграл при этом выражается через квадратурную формулу:

$$y_{n+1} = y_n + \int_{t_0}^{t_{n+1}} f(y, x)dx \approx$$

$$y_n + \int_{t_0}^{t_{n+1}} L_{n-1}(y, x)dx \approx$$

$$y_n + h\sum_{r=1}^{n} \alpha_r F(t_{i+1-r}, y_{i+1-r}),$$

Схема Адамса-Бэшфорда

k				
2	3/2	-1/2		
3	23/12	-16/12	5/12	
4	55/24	-59/24	37/24	-9/24

Схема 2-шагового алгоритма получается при использовании линейной экстраполяции по 2-м предыдущим точкам

Схема 4-шагового алгоритма получается при использовании кубической экстраполяции

Метод Булишера-Штера.

Основная идея: Метод строит рациональную интерполирующую функцию, которая в точке h/2 проходит через состояние системы после двух таких шагов, в точке h/4 проходит через состояние системы после четырех таких шагов, и т. д., а затем вычисляет значение этой функции в точке h = 0, проводя экстраполяцию. Гладкость правых частей приводит к тому, что вычисленное при помощи экстраполяции состояние системы оказывается очень близко к действительному, а использование рациональной экстраполяции вместо полиномиальной позволяет повысить точность.

После проведения одного шаг принимается решение, следует ли изменять шаг, а если да, то в какую сторону.

Считается, что метод Булирша— Штера часто оказывается более эффективным для поиска гладких решений, чем метод Рунге-Кутты.

Что такое жесткие системы ОДУ?

В данном курсе ограничимся следующим понятием жесткости системы: система будет жесткой, если на всем промежутке интегрирования с помощью явных схем необходимо вести интегрирование с очень малым шагом.

Жесткость системы проявляется тогда, когда длина промежутка интегрирования, T, удовлетворяет соотношению :

$$T \geq {}^{1}/_{|\lambda max|}$$

где λ_{max} — наибольшее по абсолютной величине собственное число матрицы системы, записанной в виде: $y' = A \cdot y$.

Алгоритмы решения для жестких систем ОДУ

Алгоритм Розенброка - является одношаговым и явным. Однако при пересчет каждого шага требуется:

- во-первых, численное определение производных функции правых частей (в случае системы ОДУ правая часть - это вектор);
- во-вторых, решения системы линейных уравнений (поскольку искомые компоненты вектора y_{n+1} входят в матричное уравнение в линейной комбинации).

Алгоритмы решения для жестких систем ОДУ

В Mathcad реализованы так же:

- алгоритм RADAU5 – алгоритм решения систем ОДУ любого типа (в том числе и жестких) основанный на неявном методе Рунге-Кутты 5-го порядка с помощью квадратур Радау
- адаптированный для жестких систем метод Булишера-Штера
- алгоритм BDF – метод обратного дифференцирования

Решения ОДУ в Mathcad

Odesolve

- предназначенная для решения дифференциальных уравнений, **линейных** относительно старшей производной.

Odesolve решает для записанных в общепринятом в математической литературе виде дифференциальных уравнений:

$$a(x)y^{(n)}+F(x,y,y',..y^{(n-1)})=f(x)$$

задачу Коши $y(x_0)=y_0$, $y'(x_0)=y_{0,1}$, $y''(x_0)=y_{0,2}$, ..., $y^{(n-1)}(x_0)=y_{0,n-1}$

или простейшую граничную задачу $y^{(k)}(a)=y_{a,k}$ $y^{(m)}(b)=y_{b,k}$, $0<= k<= n-1$, $0<= m<= n-1$.

Функция **Odesolve** по умолчанию решает поставленную задачу гибридным решателем: метод Адамса/метод обратного дифференцирования

Для решения задачи методом Рунге-Кутты с фиксированным шагом нужно щелкнуть в рабочем документе по имени функции правой кнопкой мыши и пометить во всплывающем меню пункт *Fixed.*

Для решения задачи методом Рунге-Кутты автоматическим выбором шага пометить во всплывающем меню пункт *Adaptive*.

Для решения жестких систем во всплывающем меню необходимо выбрать пункт *Radau*.

Обращение к функции для решения одного уравнения имеет вид :

Given

Формулировка уравнений и начальных/граничных условий

Y:=Odesolve(x,b,step)

Y - имя функции, содержащей значения найденного решения,

x — переменная интегрирования,

b — конец промежутка интегрирования,

step — шаг, который используется при интегрировании уравнения. Данный параметр необязателен и может быть опущен.

При вводе уравнения и условий задачи используется знак символьного равенства (**<Ctrl>+<=>**).

Для записи производных можно использовать:

- оператор дифференцирования из меню математического анализа.
- знак производной, например, вторую производную можно вводить в виде *y''(x)* (штрих записывается как **<Ctrl>+<F7>**).

<u>В любом случае необходимо обязательно записывать аргумент искомой функции.</u>

$t0 := \frac{\pi}{2}$ Начальная точка промежутка интегрирования

$T := t0 + 1$ Конечная точка промежутка интегрирования

$y0 := 0$ Значение функции в начальной точке промежутка интегрирования

$f(t,y) := \frac{y}{\tan(t)} + 2 \cdot t \cdot \sin(t)$ Правая часть ОДУ

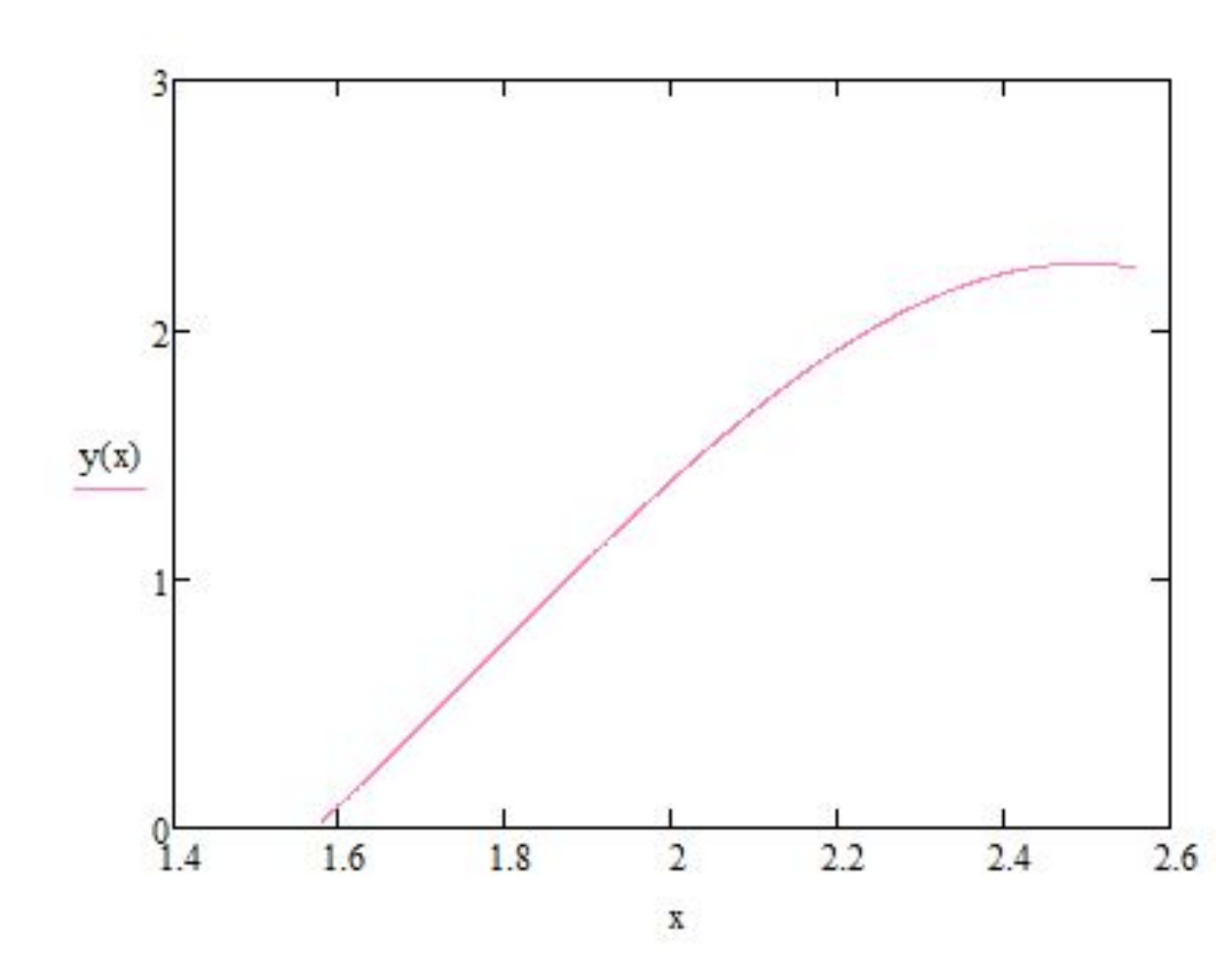
Given Ключевое слово начала блока решения

$y'(x) = f(x, y(x))$ Запись ОДУ

$y(t0) = y0$ Запись граничного условия

$y := \text{Odesolve}(x, T)$ Присвоение результатов решения ОДУ

Вывод решения ОДУ на график при использовании блока Given-Odesolve



В Mathcad решить задачу Коши для не жесткой системы ОДУ можно так же с помощью следующих функций:

- **Adams(y, x1, x2, npoints, D, {acc})** —решение задачи на отрезке методом Адамса;
- **rkfixed(y, x1, x2, npoints, D)** —решение задачи на отрезке методом Рунге—Кутты с постоянным шагом;
- **Rkadapt(y, x1, x2, npoints, D)** —решение задачи на отрезке методом Рунге—Кутты с автоматическим выбором шага;
- **Bulstoer(y, x1, x2, npoints, D)** —решение задачи на отрезке методом Булирша-Штера .

В Mathcad решить задачу Коши для жестких ОДУ можно с помощью функций:

- **BDF(y, x1, x2, npoints, D, {J},{acc})** —решение задачи на отрезке методом обратного дифференцирования (backward differentiation formula);
- **Radau(y, x1, x2, npoints, D, {J},{M},{acc}*)*** — решение задачи Radau5;
- **Stiffr(y, x1, x2, D, AJ)** — решение задачи для жестких систем на отрезке с использованием алгоритма Розенброка;
- **Stiffb(y, x1, x2, D, AJ)** —решение задачи для жестких систем на отрезке с использованием алгоритма Булирша—Штера;

В Mathcad с 14 версии существует гибридный метод:

AdamsBDF(y, x1, x2, npoints, D, {J},{acc})

В функции определяется тип системы :

жесткая/нежесткая

и выбирается соответственный решатель.

Переключение осуществляется автоматически.

- **y** — вектор начальных условий ;
- **x1**, **x2** — начальная и конечная точки отрезка интегрирования системы; для функций, вычисляющих решение в заданной точке, **x1** — начальная точка, **x2** — заданная точка;
- **npoints** — число узлов на отрезке [**x1, x2**]; при решении задачи на отрезке результат содержит **npoints+1** строку;
- **D** — имя вектор-функции **D(x,y)** правых частей, записанное через элементы искомого вектора;

- **J** — имя матрицы-функции **J(x,y)** размерности **n x (n+1)**, в первом столбце которой хранятся выражения частных производных по x правых частей системы, а в остальных n столбцах содержится матрица Якоби правых частей:

$$J(x,y)=\begin{bmatrix} \frac{\partial f_1(x,y)}{\partial x} & \frac{\partial f_1(x,y)}{\partial y_1} & \cdots & \frac{\partial f_1(x,y)}{\partial y_n} \\ \frac{\partial f_2(x,y)}{\partial x} & \frac{\partial f_2(x,y)}{\partial y_1} & \cdots & \frac{\partial f_2(x,y)}{\partial y_n} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial f_n(x,y)}{\partial x} & \frac{\partial f_n(x,y)}{\partial y_1} & \cdots & \frac{\partial f_n(x,y)}{\partial y_n} \end{bmatrix}$$

Может быть вычислен с помощью функции **Jacob (D(x,y),x)**

- **acc** — параметр, контролирующий погрешность решения при автоматическом выборе шага интегрирования (если погрешность решения больше **acc**, то шаг сетки уменьшается; шаг уменьшается до тех пор, пока его значение не станет меньше save);
- **AJ** — матрица, в которой слиты два Якобиана- по аргументу и по неизвестной функции **augment** (**Jacob (D(x,y),x), Jacob (D(x,y),y)**);

Результат работы функций :

матрица, содержащая **n+1** строк;

ее первый столбец содержит координаты узлов сетки,

второй столбец — вычисленные приближенные значения решения $\mathbf{y_1(x)}$ в узлах сетки,

(k+1) -й — значения решения $\mathbf{y_k(x)}$ в узлах сетки.

При использовании функций библиотеки **DES** (Differential Equation Solving) дифференциальные уравнения, входящие в систему, должны иметь первый порядок (то есть содержать только первые производные).

Все уравнения должны быть предварительно разрешены относительно производных и записаны в нормальной форме вида:

$$\frac{dx_i}{dt} = F_i; \quad i = \overline{1, N}$$

Для *преобразования уравнений в нормальную форму* есть два основных подхода:

- Понижение порядка уравнений путем *замены переменных*.
- Приведение системы дифференциальных уравнений к явному виду путем ее решения относительно производных

$$\begin{cases} y''+\alpha y'+\beta y+\gamma=0 \\ y(x_1)=y1 \\ y(x_2)=y2 \end{cases} \qquad x\in[x_1,x_2]$$

1

$$\begin{cases} y'=p(x) \\ p'=-\alpha p(x)-\beta y-\gamma \\ y(x_1)=y1 \\ y(x_2)=y2 \end{cases}$$

2

$$X=\begin{pmatrix} y \\ p \end{pmatrix}=\begin{pmatrix} X_0 \\ X_1 \end{pmatrix}$$

3

$$D(t,X)=\begin{pmatrix} X_1 \\ -\alpha X_1-\beta X_0-\gamma \end{pmatrix}$$

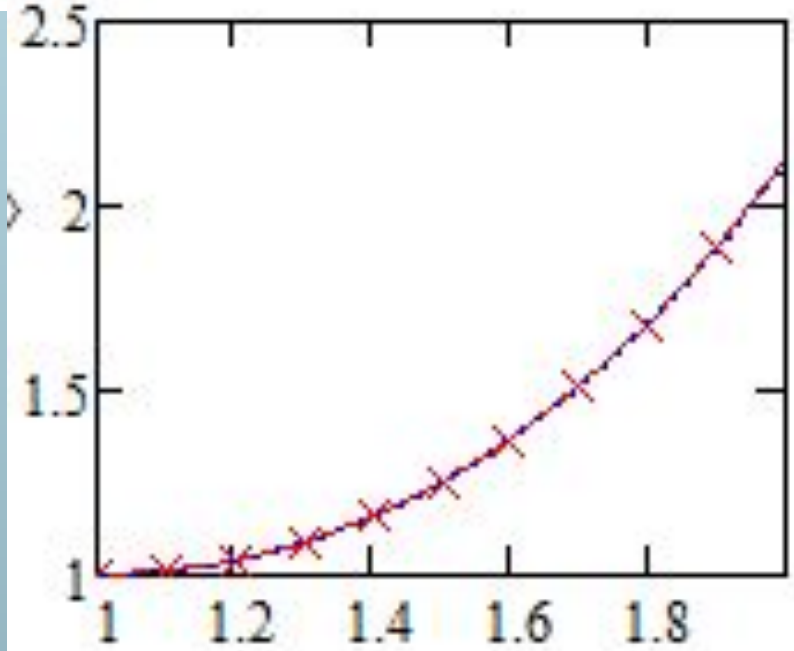
Given

$$y''(x) + \frac{1}{x} \cdot y'(x) = f(x, y(x))$$

$$y(1) = 1 \qquad y'(1) = 0$$

$$\begin{cases} y'(x) = p(x) \\ p'(x) = f(x, y(x)) - \dfrac{p(x)}{x} \end{cases}$$

m := Odesolve(x, 2)



$$D(t, X) := \begin{pmatrix} X_1 \\ f(t, X_0) - \dfrac{X_1}{t} \end{pmatrix}$$

$$m := \text{rkfixed}\left[\begin{pmatrix} 1 \\ 0 \end{pmatrix}, 1, 2, 100, D\right]$$

Решение краевых задач.

Решение краевых задач для систем ОДУ методом стрельбы в Mathcad достигается применением двух встроенных функций. Первая предназначена для двухточечных задач с краевыми условиями, заданными на концах интервала:

sbval (z,x0,x1, D, load, score)

— поиск вектора недостающих **L** начальных условий для двухточечной краевой задачи для системы **N** ОДУ.

z — вектор размера **Lx1**, присваивающий недостающим начальным условиям (на левой границе интервала) начальные значения;

x0 — левая граница расчетного интервала;

x1 — правая граница расчетного интервала;

load(x0,z) — векторная функция размера **Nx1** левых граничных условий, причем недостающие начальные условия поименовываются соответствующими компонентами векторного аргумента **z**;

score (x1, y) — векторная функция размера **Lx1**, выражающая **L** правых граничных условий для векторной функции y в точке **x1**;

D(x,y) — векторная функция, описывающая систему **N** ОДУ, размера **Nx1** и двух аргументов — скалярного **x** и векторного y. При этом y — это неизвестная векторная функция аргумента x того же размера **Nx1**.

Решение краевых задач в Mathcad реализовано не совсем очевидным образом.

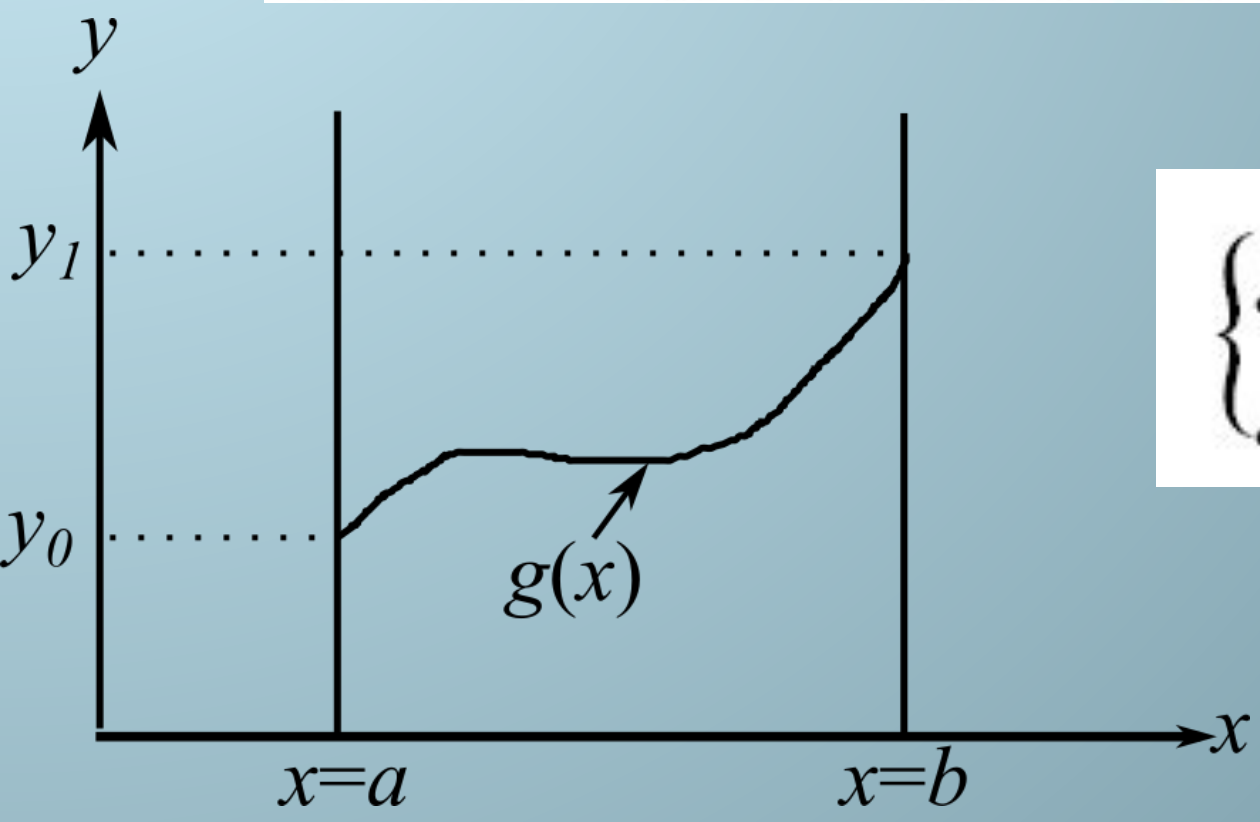
Необходимо помнить, что число элементов векторов x0 и ***load*** *равно количеству уравнений N, а векторов z, score и результата действия функции* ***sbval****— количеству правых граничных условий L. Соответственно, левых граничных условий в задаче должно быть* ***(N-L)*** *.*

Начальное значение фактически является параметром численного метода и поэтому может решающим образом повлиять на решение краевой задачи.

ВНИМАНИЕ! Метод стрельбы не годится для решения жестких краевых задач. Поэтому алгоритмы решения жестких ОДУ в Mathcad приходится программировать самому

Краевая задача

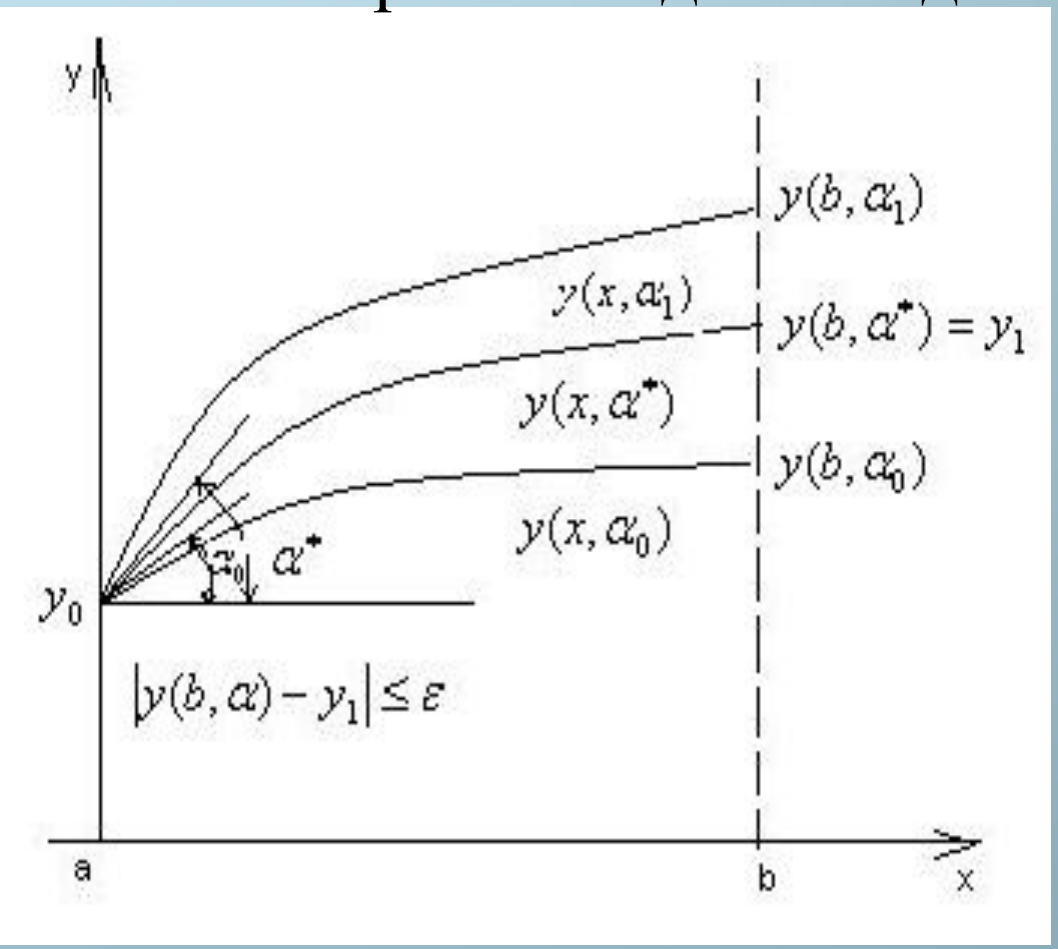
$$g''(x) = f(x, g, g'), \quad a < x < b;$$



$$\begin{cases} g|_{x=a} = y_0 \\ g|_{x=b} = y_1 \end{cases}$$

Краевая задача.
Алгоритм стрельбы.

Краевая задача сводится к задаче Коши:



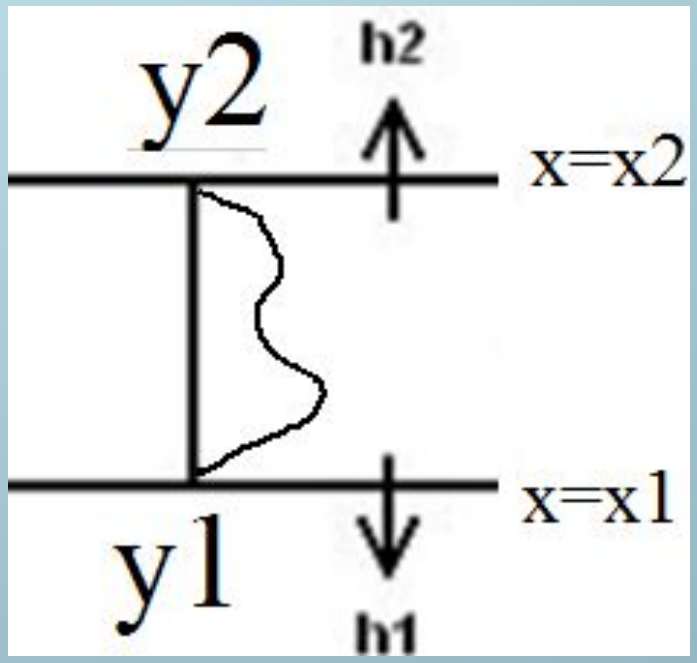
$y(x, \alpha)'' = f(x,\alpha,y,y'), \quad a < x < b;$

$$\begin{cases} y(x,\alpha)|_{x=a} = y_0 \\ y'(x,\alpha)|_{x=a} = \alpha \end{cases}$$

$$y(x,\alpha)|_{x=b} = y_1$$

$$\begin{cases} y''+xy'=0 \\ y(x_1)=y1 \\ y(x_2)=y2 \end{cases}$$

$$\begin{cases} y'=p(x) \\ p'=-y'\cdot x=-p(x)\cdot x \\ y(x_1)=y1 \\ y(x_2)=y2 \end{cases}$$



$$X=\begin{pmatrix} y \\ p \end{pmatrix}=\begin{pmatrix} X_0 \\ X_1 \end{pmatrix}$$

$$D(t,X)=\begin{pmatrix} X_1 \\ t\cdot X_1 \end{pmatrix}$$

----------Задание параметров задачи----------

$y0 := 1.1$ значение функции на левой границе

$yk := 10.1$ значение функции на правой границе

$x0 := 1$ значение начальной координаты

$xk := 2$ значение конечной координаты

----------Решение задачи----------

$N := 150$ определяем число точек, в которых будет найдено численное решение

$f1(x) := x$ определяем вспомогательную функцию для вектора правых частей

$$D(t,X) := \begin{pmatrix} X_1 \\ -f1(t)\cdot X_1 \end{pmatrix}$$ определяем вектор правых частей

$z_0 := 2$ начальное приближение для метода стрельбы

$load(x0, z) := \begin{pmatrix} y0 \\ z_0 \end{pmatrix}$ определение векторной функции левых граничных условий

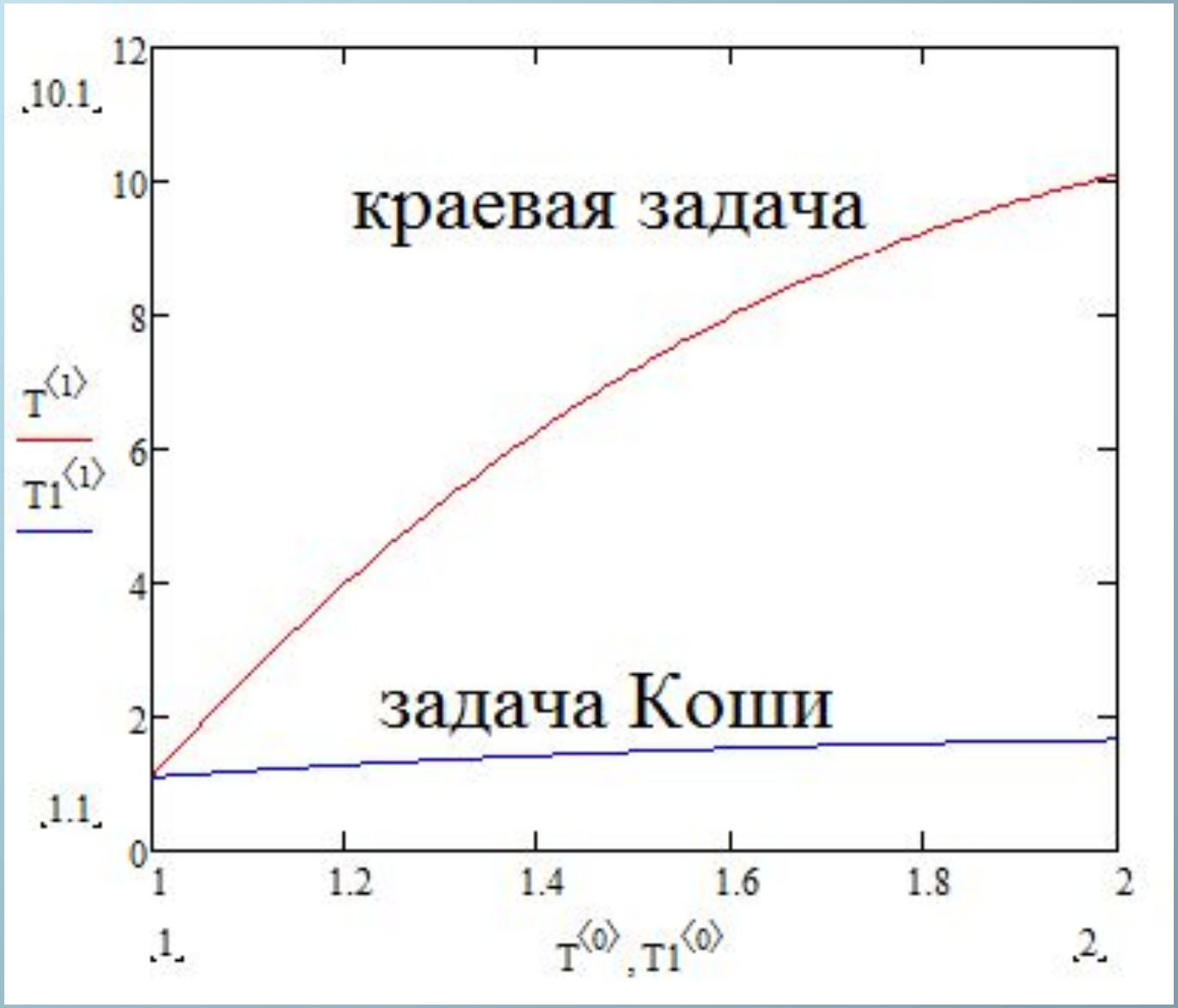
$score(x1, X) := X_0 - yk$ определение векторной функции для определения граничных условий на правой границе

$I1 := sbval(z, x0, xk, D, load, score)$ поиск недостающих граничных условий на левой границе

$I1 = (16.024)$

$T := rkfixed\left[\begin{pmatrix} y0 \\ I1_0 \end{pmatrix}, x0, xk, N, D\right]$ находим численное решение краевой задачи для исходного дифференциального уравнения на заданном промежутке в заданном числе точек

$T1 := rkfixed\left[\begin{pmatrix} y0 \\ 1 \end{pmatrix}, x0, xk, N, D\right]$ находим численное решение задачи Коши для исходного дифференциального уравнения на заданном промежутке в заданном числе точек


краевая задача
задача Коши
12
10
8
6
4
2
0
10.1
1.1
1
1.2
1.4
1.6
1.8
2
T(1)
T1(1)
T(0) , T1(0)
1
2

- **z1** — вектор, присваивающий недостающим начальным условиям на левой границе интервала начальные значения.
- **z2** — вектор того же размера, присваивающий недостающим начальным условиям на правой границе интервала начальные значения.
- **x0** — левая граница расчетного интервала.
- **x1** — правая граница расчетного интервала.
- **xf** — точка внутри интервала.
- **Do(x,y)** — векторная функция, описывающая систему **N** ОДУ размера **Nx1** и двух аргументов — скалярного **x** и векторного y. При этом y — это неизвестная векторная функция аргумента **x** того же размера **Nx1**.
- **load1(x0,z)** — векторная функция размера **Nx1** левых граничных условий, причем недостающие начальные условия поименовываются соответствующими компонентами векторного аргумента **z**.
- **load2(x1,z)** — векторная функция размера **Nx1** левых граничных условий, причем недостающие начальные условия поименовываются соответствующими компонентами векторного аргумента **z**.
- **score (xf, y)** — векторная функция размера **Nx1**, выражающая внутреннее условие для векторной функции y в точке **xf**.