Краевая задача для ОДУ

Цыбулин Иван (tsybulin@crec.mipt.ru)

Линейная краевая задача для ОДУ

$$\frac{d\mathbf{y}(x)}{dx} = \mathbf{A}(x)\mathbf{y}(x) + \mathbf{f}(x), \qquad x \in [a, b]$$

$$\ell_i^\top \mathbf{y}(a) = \alpha_i, \quad i = 1, \dots, r$$

$$\ell_i^\top \mathbf{y}(b) = \alpha_i, \quad i = r + 1, \dots, n$$

Построение фундаментальной системы

Решение ОДУ зависит от n констант, причем в линейном случае можно записать

$$\mathbf{y}(x) = \mathbf{y}_0(x) + \sum_{i=1}^{n} C_i \mathbf{y}_i(x),$$

где $\mathbf{y}_0(x)$ — частное решение ОДУ, $\mathbf{y}_i(x)$ — линейно независимые решения однородного уравнения

$$\frac{d\mathbf{y}(x)}{dx} = \mathbf{A}(x)\mathbf{y}(x), \qquad x \in [a, b]$$

Фундаментальную систему можно построить численно. Для этого можно найти каждое $\mathbf{y}_i(x)$, сформулировав для нее подходящую задачу Коши. Например,

$$\begin{cases} \dfrac{d\mathbf{y}_0(x)}{dx} &= \mathbf{A}(x)\mathbf{y}_0(x) + \mathbf{f}(x) \\ \mathbf{y}_0(a) &= \mathbf{0} \end{cases} \qquad \begin{cases} \dfrac{d\mathbf{y}_i(x)}{dx} &= \mathbf{A}(x)\mathbf{y}_i(x) \\ \mathbf{y}_i(a) &= \mathbf{e}_i \end{cases}$$

Решение краевой задачи

После нахождения фундаментальной системы, решение краевой задачи сводится к разрешению краевых условий относительно C_i:

$$\ell_i^\top \left[\mathbf{y}_0(a) + \sum_{i=1}^{n} C_i \mathbf{y}_i(a) \right] = \alpha_i, \quad i = 1, \ldots, r$$

$$\ell_i^\top \left[\mathbf{y}_0(b) + \sum_{i=1}^{n} C_i \mathbf{y}_i(b) \right] = \alpha_i, \quad i = r+1, \ldots, n$$

Это линейная система из n уравнений для n неизвестных C_i.

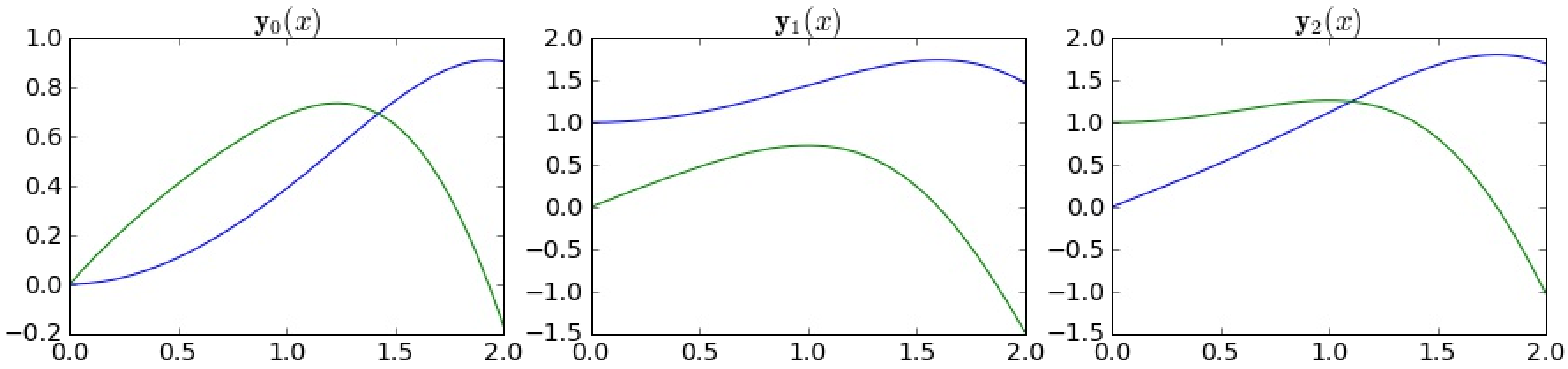
Решим численно систему

$$
\begin{aligned}
u'(x) &= v(x) \\
v'(x) &= (1 - x^2)u(x) + e^{-x}
\end{aligned}
$$

$$
u(0) = 1, \qquad u(2) + v(2) = 0
$$

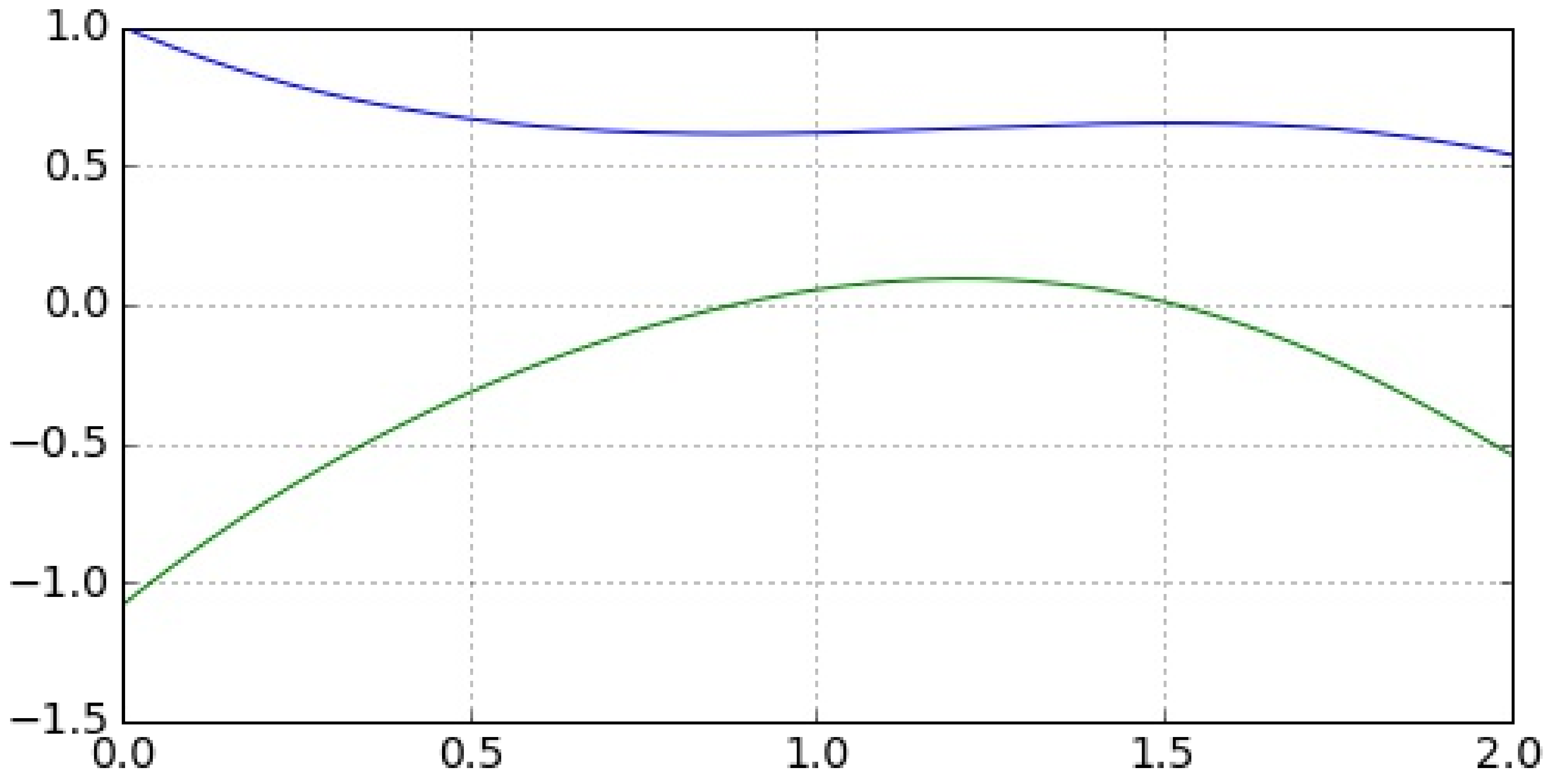
```
def G(x, y): # Правая часть неоднородной ОДУ
    u, v = y; return np.array([v, (1-x*x)*u + np.exp(-x)])
def Ghom(x, y): # Правая часть однородной ОДУ
    u, v = y; return np.array([v, (1-x*x)*u])
a = 0; b = 2
X, Y0 = fixed_stepsize(G   , np.array([0., 0.]), b, rk4, b / 200, verbose=True)
X, Y1 = fixed_stepsize(Ghom, np.array([1., 0.]), b, rk4, b / 200)
X, Y2 = fixed_stepsize(Ghom, np.array([0., 1.]), b, rk4, b / 200)

plt.figure(figsize=(15, 3)); plt.rc('font', size=14)
plt.subplot(1, 3, 1); plt.plot(X, Y0); plt.title(r'${\bf y}_0(x)$')
plt.subplot(1, 3, 2); plt.plot(X, Y1); plt.title(r'${\bf y}_1(x)$')
plt.subplot(1, 3, 3); plt.plot(X, Y2); plt.title(r'${\bf y}_2(x)$')
plt.show()
```

Классический метод РК 4 порядка, всего шагов: 200



```python
# Теперь можно разрешить условия u(0) = 1, u(2) + v(2) = 0
# (u(x),v(x)) = Y0(x) + C1 Y1(x) + C2 Y2(x)

A = [[      Y1[0, 0],             Y2[0, 0]       ], # Левое условие
     [Y1[-1, 0]+Y1[-1, 1], Y2[-1, 0]+Y2[-1, 1]]] # Правое условие
b = np.array([1, 0]) - [Y0[0, 0], Y0[-1, 0]+Y0[-1, 1]]
C1, C2 = np.linalg.solve(A, b)
print('C1 = %f, C2 = %f' % (C1, C2))
plt.figure(figsize=(8, 4)); plt.plot(X, Y0 + C1*Y1 + C2*Y2); plt.grid()
```

C1 = 1.000000, C2 = -1.082561



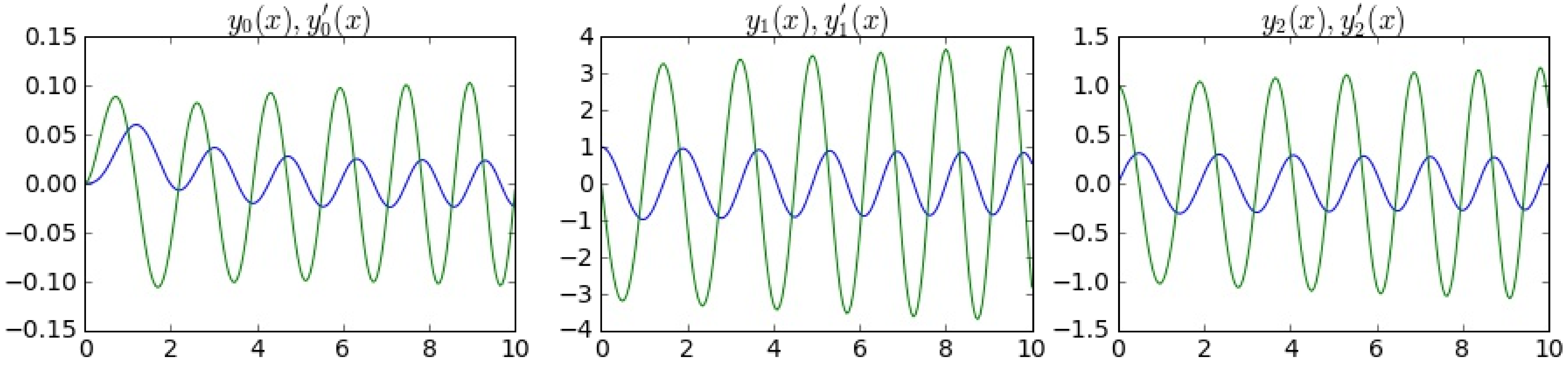
Задача XI.8.1

Построить общее решение для

- $y''(x) + (10 + x)y(x) = xe^{-x}, \quad x \in [0, 10]$
- $y''(x) - (10 + x)y(x) = xe^{-x}, \quad x \in [0, 10]$

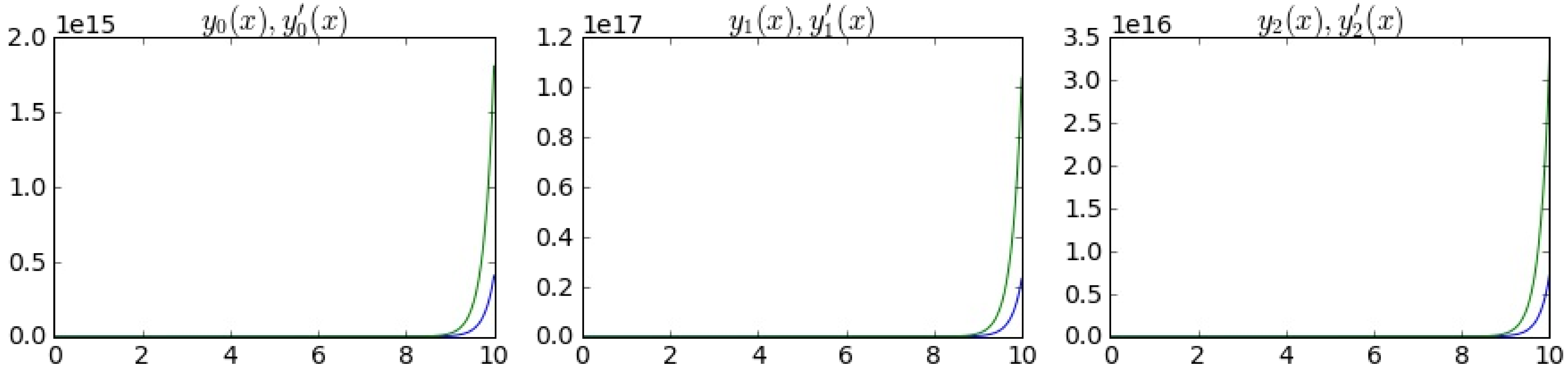
```
def G(x, y): # Правая часть неоднородной ОДУ
    u, v = y; return np.array([v, -(10+x)*u + x*np.exp(-x)])
def Ghom(x, y): # Правая часть однородной ОДУ
    u, v = y; return np.array([v, -(10+x)*u])

a = 0; b = 10
X, Y0 = fixed_stepsize(G   , np.array([0., 0.]), b, rk4, b / 200, verbose=True)
X, Y1 = fixed_stepsize(Ghom, np.array([1., 0.]), b, rk4, b / 200)
X, Y2 = fixed_stepsize(Ghom, np.array([0., 1.]), b, rk4, b / 200)

plt.figure(figsize=(15, 3)); plt.rc('font', size=14)
plt.subplot(1, 3, 1); plt.plot(X, Y0); plt.title(r'$y_0(x), y_0^\prime(x)$')
plt.subplot(1, 3, 2); plt.plot(X, Y1); plt.title(r'$y_1(x), y_1^\prime(x)$')
plt.subplot(1, 3, 3); plt.plot(X, Y2); plt.title(r'$y_2(x), y_2^\prime(x)$')
plt.show()
```

Классический метод РК 4 порядка, всего шагов: 200



```python
def G(x, y): # Правая часть неоднородной ОДУ
    u, v = y; return np.array([v, (10+x)*u + x*np.exp(-x)])
def Ghom(x, y): # Правая часть однородной ОДУ
    u, v = y; return np.array([v, (10+x)*u])

a = 0; b = 10
X, Z0 = fixed_stepsize(G   , np.array([0., 0.]), b, rk4, b / 200, verbose=True)
X, Z1 = fixed_stepsize(Ghom, np.array([1., 0.]), b, rk4, b / 200)
X, Z2 = fixed_stepsize(Ghom, np.array([0., 1.]), b, rk4, b / 200)

plt.figure(figsize=(15, 3)); plt.rc('font', size=14)
plt.subplot(1, 3, 1); plt.plot(X, Z0); plt.title(r'$y_0(x), y_0^\prime(x)$')
plt.subplot(1, 3, 2); plt.plot(X, Z1); plt.title(r'$y_1(x), y_1^\prime(x)$')
plt.subplot(1, 3, 3); plt.plot(X, Z2); plt.title(r'$y_2(x), y_2^\prime(x)$')
plt.show()
```

Классический метод РК 4 порядка, всего шагов: 200



Мы численно построили для каждой задачи фундаментальную систему в форме

$$y(x) = y_0(x) + C_1 y_1(x) + C_2 y_2(x).$$

Но можно ли использовать эту систему для решения краевой задачи? Пусть граничные условия заданы в виде

$$y(a) = \alpha,\ y(b) = \beta$$

Матрица системы для определения C_1, C_2 будет иметь вид

$$A = \begin{pmatrix} y_1(a) & y_2(a) \\ y_1(b) & y_2(b) \end{pmatrix}$$

```python
def cond(A):
    lam = np.sqrt(np.linalg.eigvalsh(A.T * A))
    return lam[-1] / lam[0]

A = np.array([[Y1[0, 0], Y2[0, 0]], [Y1[-1, 0], Y2[-1, 0]]])
print('Для колебательных решений cond(A) =', cond(A))

A = np.array([[Z1[0, 0], Z2[0, 0]], [Z1[-1, 0], Z2[-1, 0]]])
print('Для экспоненциальных решений cond(A) =', cond(A))
```

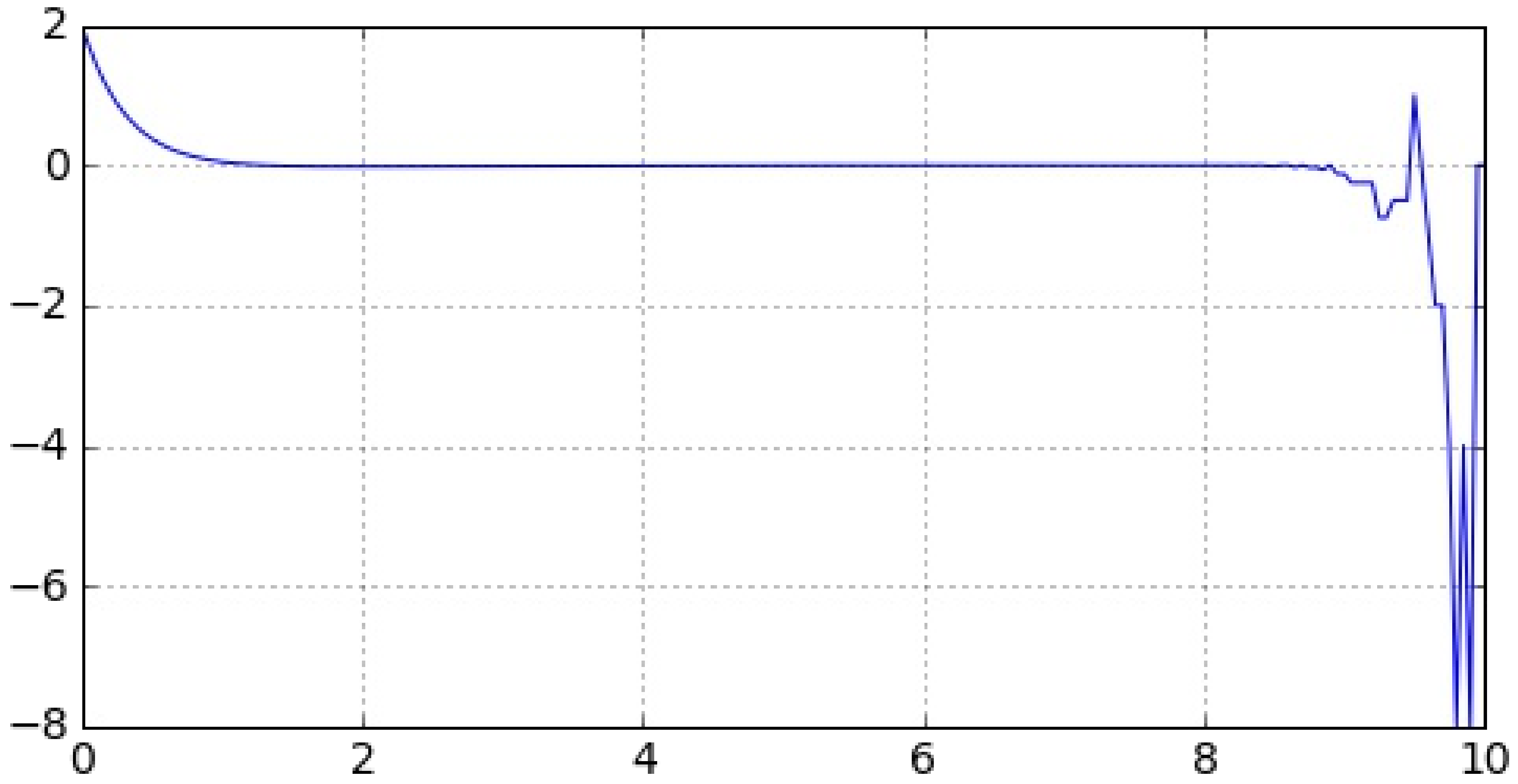
```
Для колебательных решений cond(A) = 5.01011683375
Для экспоненциальных решений cond(A) = 7.29850742218e+15
```

Видно, что во втором случае система для определения C_1, C_2 очень плохо определена, а значит C_1, C_2 будут найдены с большими ошибками, что в свою очередь приведет к большим ошибкам в итоговом решении, вычисляемом по формуле

$$y(x) = y_0(x) + C_1 y_1(x) + C_2 y_2(x)$$

```
A = np.array([[Z1[0, 0], Z2[0, 0]], [Z1[-1, 0], Z2[-1, 0]]])
b = np.array([2-Z0[0, 0], 3-Z0[-1, 0]])
C1, C2 = np.linalg.solve(A, b)
print('C1 = %f, C2 = %f' % (C1, C2))
plt.figure(figsize=(8, 4)); plt.plot(X, Z0[:, 0] + C1*Z1[:, 0] + C2*Z2[:, 0])
plt.grid()
```

C1 = 2.000000, C2 = -6.429219



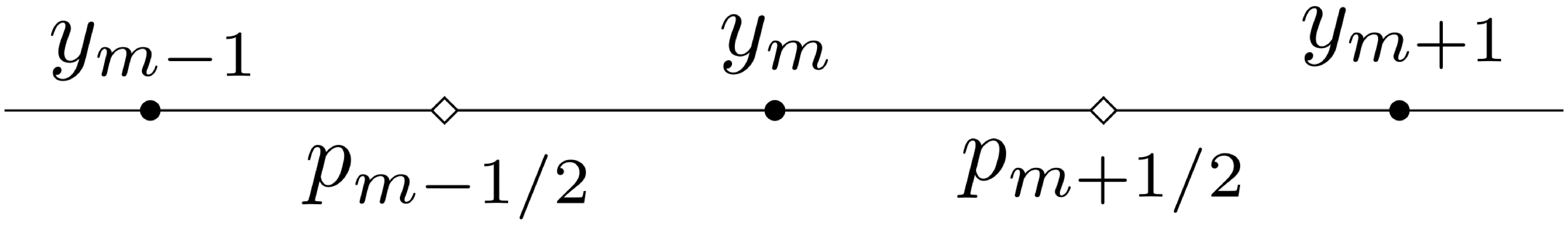
Для краевых задач для уравнений второго порядка в форме

$$\frac{d}{dx}\left[p(x)\frac{dy}{dx}\right]+r(x)\frac{dy}{dx}-q(x)y(x)=f(x)$$

можно вопользоваться другим методом — методом прогонки (конечно-разностным методом). Для этого уравнение аппроксимируется на равномерной сетке на отрезке $[a,b]$.

$$\frac{p_{m+1/2}(y_{m+1}-y_m)-p_{m-1/2}(y_m-y_{m-1})}{h^2}+r_m\frac{y_{m+1}-y_{m-1}}{2h}-q_m y_m=f_m,$$

$$p_{m+1/2}\equiv p\left(\frac{x_m+x_{m+1}}{2}\right),\quad q_m\equiv q(x_m),\quad r_m\equiv r(x_m),\quad f_m\equiv f(x_m)$$



Такая аппроксимация уравнения имеет второй порядок и обладает преимуществами перед аппроксимацией

$$\frac{d}{dx}\left[p(x)\frac{dy}{dx}\right] = \frac{dp}{dx}\frac{dy}{dx} + p(x)\frac{d^2y}{dx^2} \sim \frac{p_{m+1}-p_{m-1}}{2h}\,\frac{y_{m+1}-y_{m-1}}{2h}$$

$$+\,p_m\frac{y_{m+1}-2y_m+y_{m-1}}{h^2}.$$

Граничные условия первого рода $y(a)=\alpha, y(b)=\beta$ аппроксимируются в разностной задаче тривиально:

$$y_0=\alpha, \qquad y_M=\beta.$$

Для условий с производными

$$p(a)y'(a)+\sigma y(a)=\gamma$$

$$p(b)y'(b)+\rho y(a)=\delta$$

аппроксимация на границе со вторым порядком производится сложнее.

Рассмотрим левое условие $p(a)y'(a) + \sigma y(a) = \gamma$. Проинтегрируем уравнение

$$\frac{d}{dx}\left[p(x)\frac{dy}{dx}\right] + r(x)\frac{dy}{dx} - q(x)y(x) = f(x)$$

на отрезке $[a, a+h]$, предварительно умножив на $\phi(x) = 1 - \frac{x-a}{h}$.

$$0 = \int_a^{a+h}\left[\frac{d}{dx}\left(p(x)\frac{dy}{dx}\right) + r(x)\frac{dy}{dx} - q(x)y(x) - f(x)\right]\phi(x)dx =$$

$$= -p(a)y'(a) + \int_a^{a+h}\left(r(x)\phi(x) - p(x)\frac{d\phi}{dx}\right)\frac{dy}{dx}dx -$$

$$\int_a^{a+h}(q(x)y(x) + f(x))\phi(x)dx =$$

Приближая первый интеграл по формуле средней точки, а второй по формуле трапеций, получим

$$= -p(a)y'(a) + \left(\frac{r_{1/2}}{2} + \frac{p_{1/2}}{h}\right)\frac{y_1 - y_0}{h} - \frac{h}{2}(q_0 y_0 + f_0) + O(h^2).$$

Таким образом,

$$p(a)y'(a) = \left(\frac{r_{1/2}}{2} + \frac{p_{1/2}}{h}\right)\frac{y_1 - y_0}{h} - \frac{h}{2}(q_0 y_0 + f_0) + O(h^2).$$

Требуемая аппроксимация граничного условия

$$p(a)y'(a) + \sigma y(a) = \gamma$$

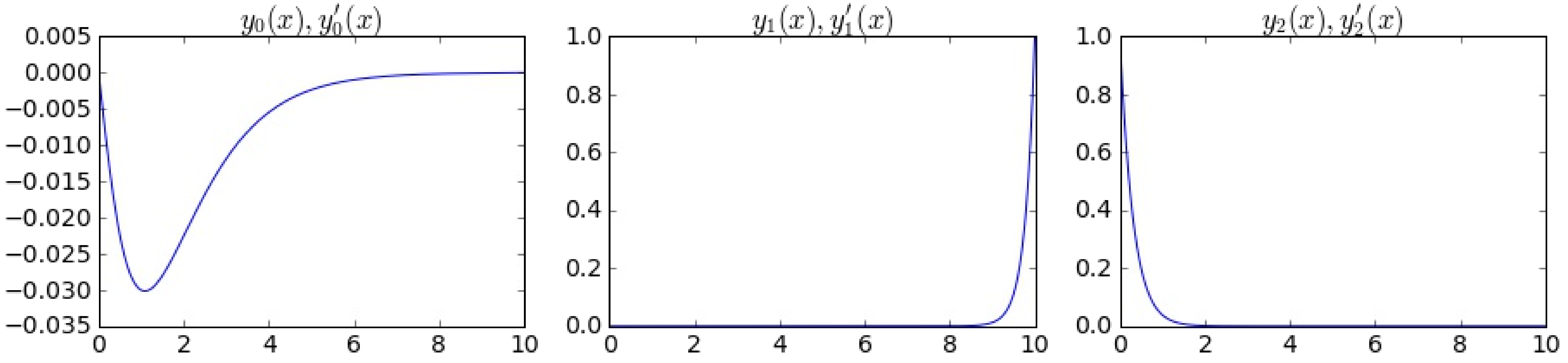
принимает вид

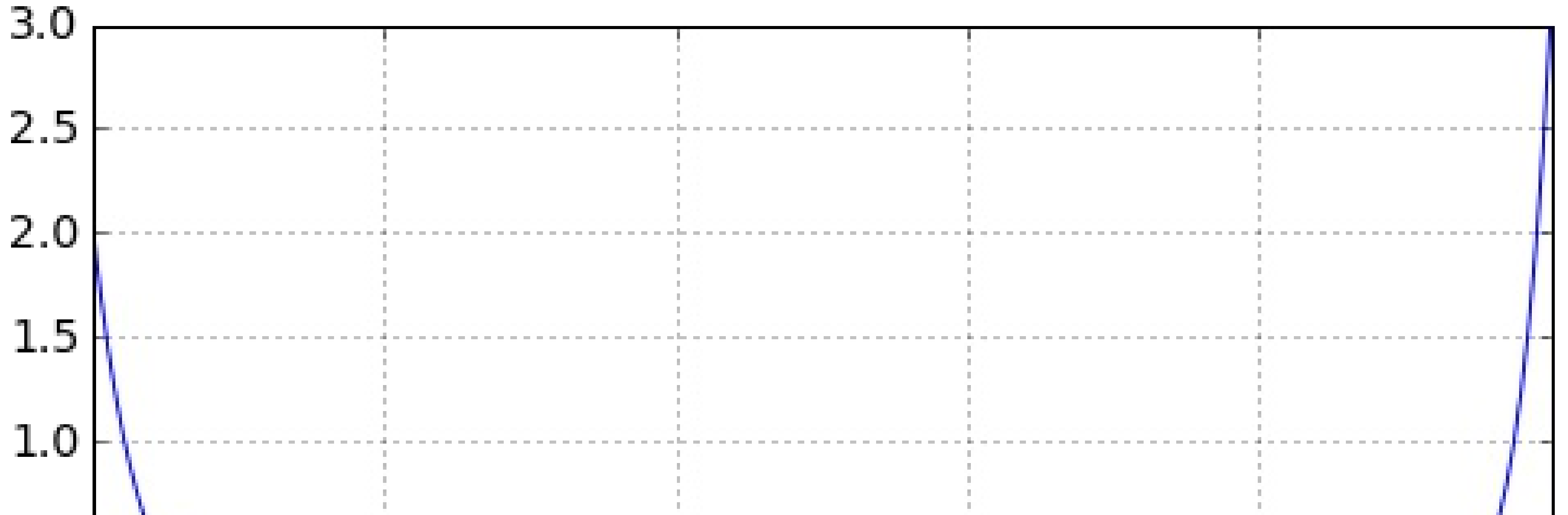
$$\left(\frac{r_{1/2}}{2} + \frac{p_{1/2}}{h}\right)\frac{y_1 - y_0}{h} - \frac{h}{2}(q_0 y_0 + f_0) + \sigma y_0 = \gamma$$

Возвращаясь к задаче XI.7.1, построим фундаментальное решение
$$y''(x) - (10 + x)y(x) = xe^{-x}, \quad x \in [0, 10]$$
как линейную комбинацию
$$y(x) = y_0(x) + C_1 y_1(x) + C_2 y_2(x),$$
где y_i получаются не из решение задачи Коши, а из решения *краевых задач* методом прогонки.
$$\begin{cases} y_0''(x) - (10 + x)y_0(x) = xe^{-x} \\ y_0(0) = 0 \\ y_0(10) = 0 \end{cases} \qquad \begin{cases} y_1''(x) - (10 + x)y_1(x) = 0 \\ y_1(0) = 0 \\ y_1(10) = 1 \end{cases}$$
$$\begin{cases} y_2''(x) - (10 + x)y_2(x) = 0 \\ y_2(0) = 1 \\ y_2(10) = 0 \end{cases}$$

```python
from scipy.linalg import solve_banded

M = 200; X = np.linspace(0, 10, M+1)
def sweep_solve(x, alpha, beta, rhs):
    h = x[1] - x[0]; M = len(x) - 1
    A = np.zeros((3, M+1)); b = np.zeros(M+1)

    # Заполняем строки трехдиагональной системы с 1 до M-1
    A[0, 2:] = A[2, :-2] = 1/h**2
    A[1, 1:M] = -2/h**2 - (10 + x[1:M])
    b[1:M] = rhs[1:M]

    # Первая и последняя строки содержат граничные условия
    A[0, 1] = A[2, -1] = 0
    A[1, 0] = A[1, M] = 1
    b[0] = alpha; b[M] = beta;
    return solve_banded((1, 1), A, b)
```

```
Z0 = sweep_solve(X, 0, 0, X*np.exp(-X))
Z1 = sweep_solve(X, 0, 1, np.zeros_like(X))
Z2 = sweep_solve(X, 1, 0, np.zeros_like(X))

plt.figure(figsize=(15, 3)); plt.rc('font', size=14)
plt.subplot(1, 3, 1); plt.plot(X, Z0); plt.title(r'$y_0(x), y_0^\prime(x)$')
plt.subplot(1, 3, 2); plt.plot(X, Z1); plt.title(r'$y_1(x), y_1^\prime(x)$')
plt.subplot(1, 3, 3); plt.plot(X, Z2); plt.title(r'$y_2(x), y_2^\prime(x)$')
plt.show()
plt.figure(figsize=(8, 4)); plt.plot(X, Z0 + 3*Z1 + 2*Z2)
plt.grid()
```





Нелинейная краевая задача. Метод стрельбы

Рассмотрим нелинейную краевую задачу второго порядка

$$y''(x) = f(x, y(x), y'(x)), \qquad x \in [a, b]$$
$$y(a) = \alpha$$
$$y(b) = \beta$$

Для нелинейных задач не существует понятия фундаментальной системы, так как решение больше не представляется *линейной* комбинацией.

Рассмотрим вспомогательную задачу Коши

$$y''(x) = f(x, y(x), y'(x)), \qquad x \in [a, b]$$

$$y(a) = \alpha$$

$$y'(a) = p.$$

Здесь p — параметр. При некотором его значении решение задачи Коши совпадает с решением исходной краевой задачи. При каждом значении параметра p задача может быть легко решена численно.

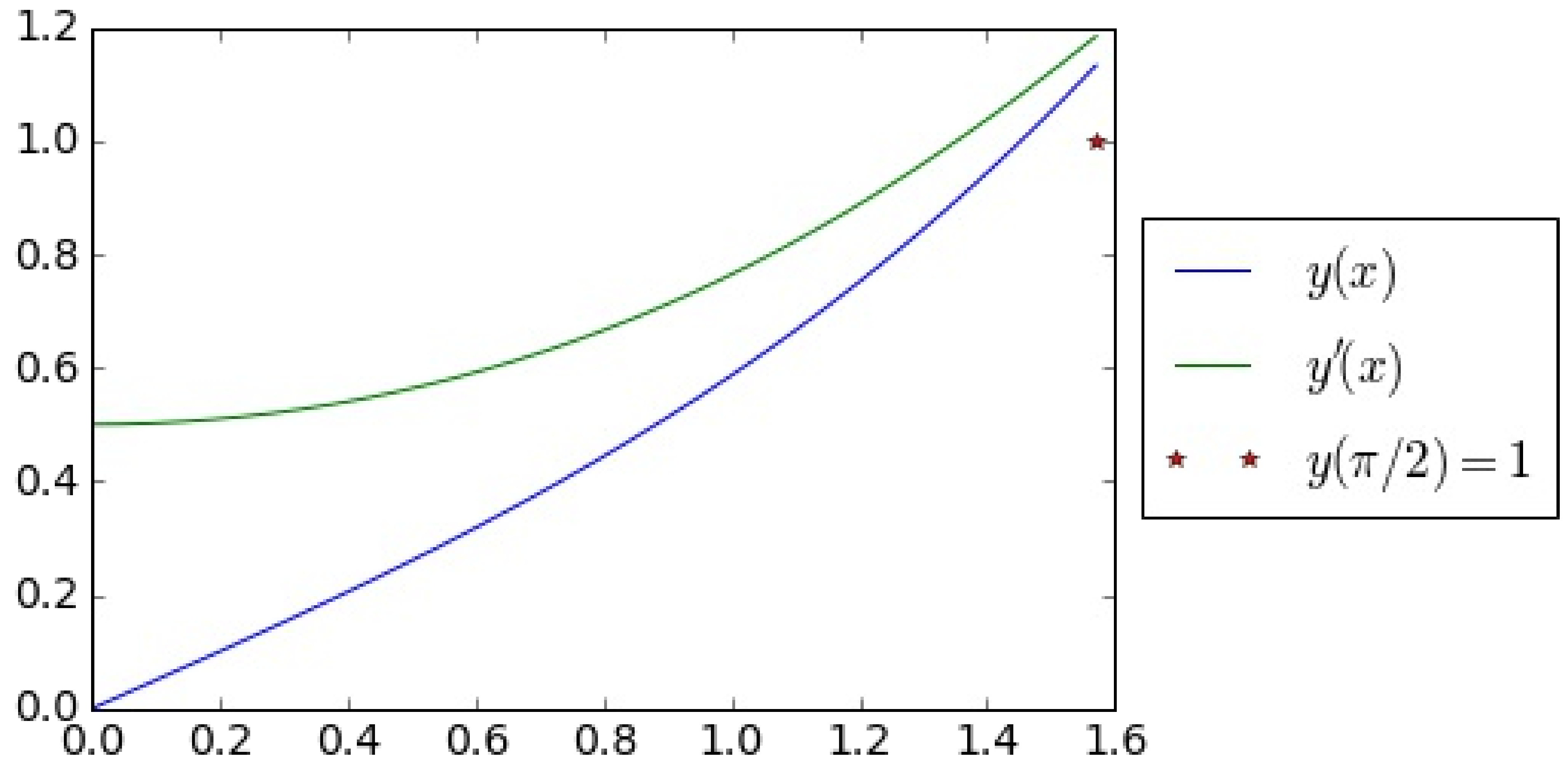
Рассмотрим на примере

$$y'' = \sin y, \quad x \in [0, \pi/2]$$

$$y(0) = 0$$

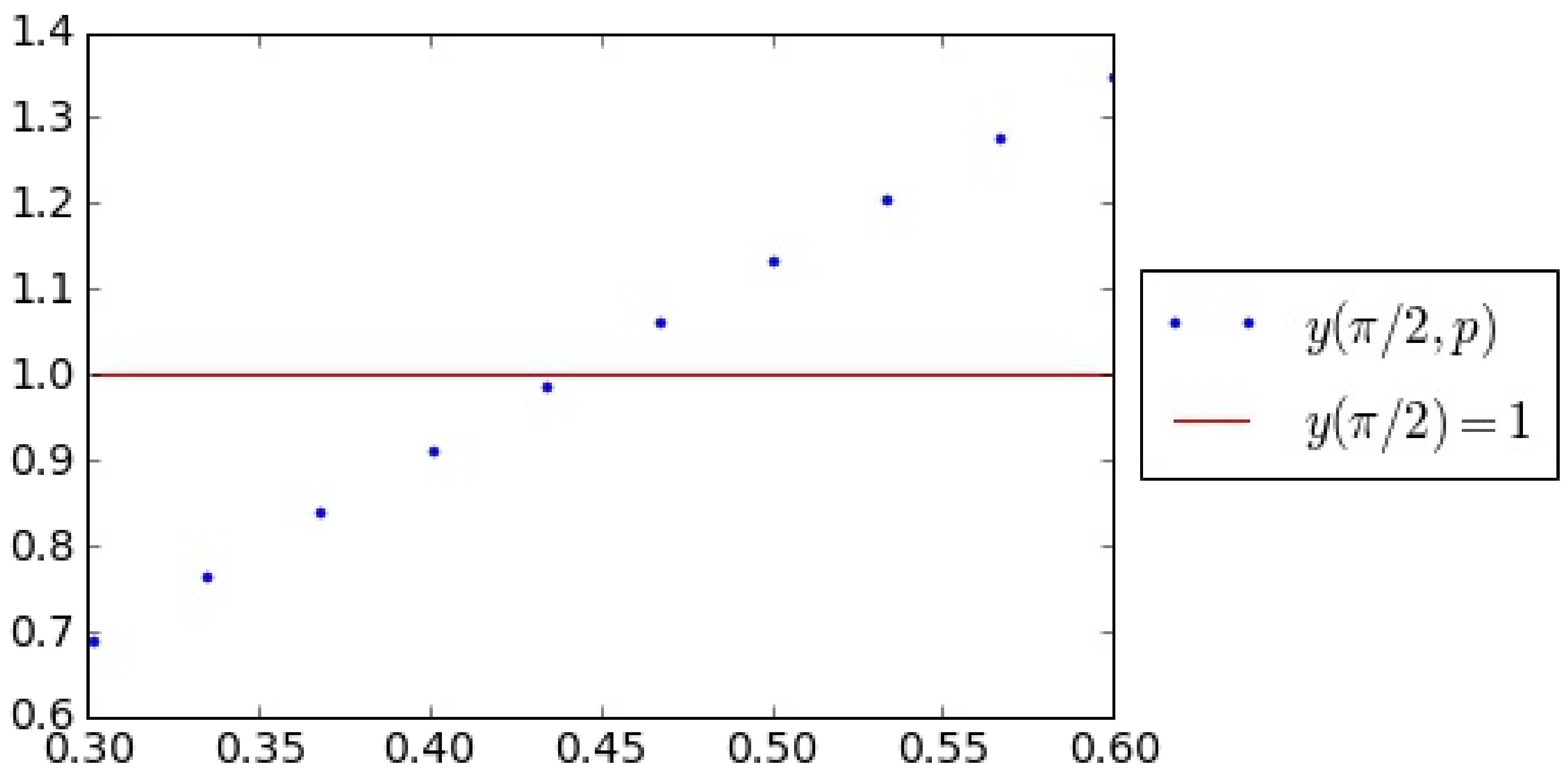
$$y(\pi/2) = 1$$

```
def G(x, Y): # Правая часть для задачи Коши
    y, yp = Y
    return np.array([yp, np.sin(y)])

def shoot_plot(p):
    b = np.pi/2
    X, Y = fixed_stepsize(G, np.array([0., p]), b, rk4, b / 200)
    plt.plot(X, Y[:, 0], label="$y(x)$")
    plt.plot(X, Y[:, 1], label="$y'(x)$")
    plt.plot([np.pi/2], [1], '*', label='$y(\pi/2) = 1$')
    plt.legend(loc='center left', bbox_to_anchor=(1., .5))
    return

p_approx = 0.5
shoot_plot(p_approx)
```



Значение решения на правом конце можно рассмотреть как функцию p. Мы умеем вычислять ее поточечно при каждом p.

```
def shoot(p):
    b = np.pi/2
    X, Y = fixed_stepsize(G, np.array([0., p]), b, rk4, b / 200)
    return Y[-1, 0]

val = []; P = np.linspace(0.301, 0.6, 10)
for p in P:
    val.append(shoot(p))

plt.plot(P, val, '.', label='$y(\pi/2, p)$');
plt.plot(P, np.ones_like(P), 'r', label='$y(\pi/2) = 1$');
plt.xlabel('$p$'); plt.legend(loc='center left', bbox_to_anchor=(1., .5))
plt.show()
```

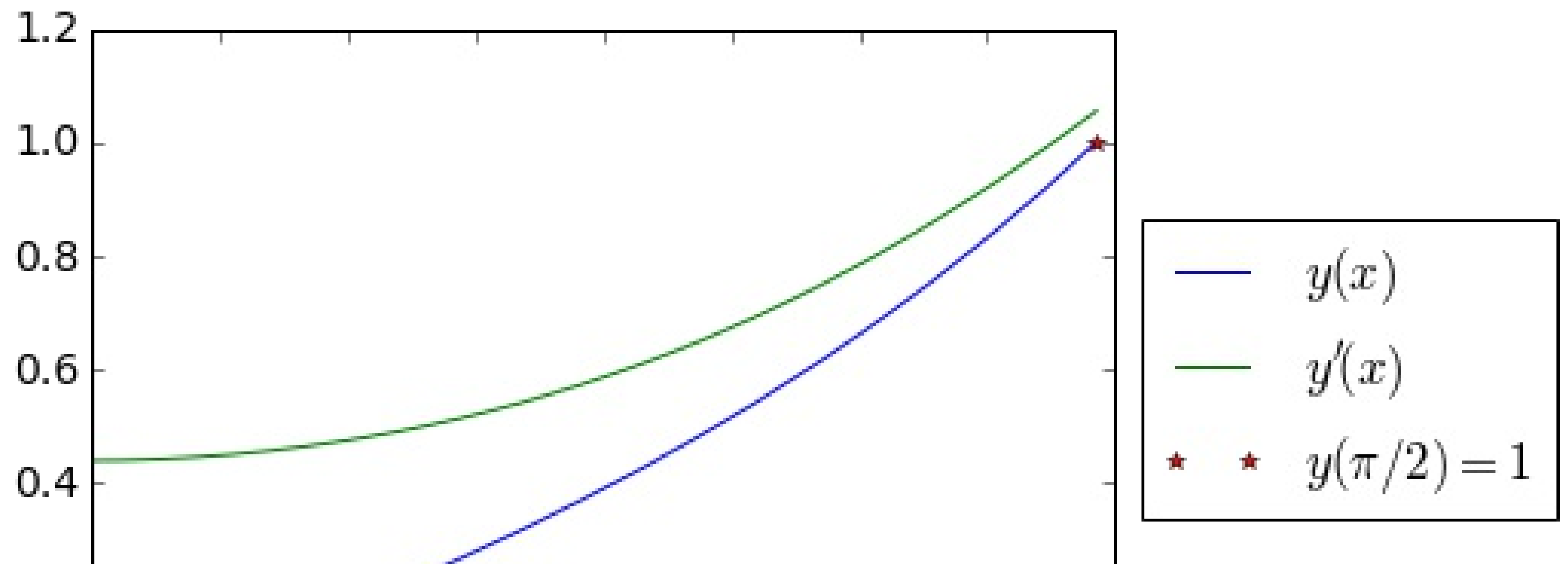


Задача определения p свелась к решению некоторого нелинейного уравнения $F(p) = 1$, где сама функция $F(p) = y(\pi/2, p)$ задана достаточно сложно — как решение задачи Коши. Однако, это не мешает применить к ней численные методы для поиска корней нелинейных уравнений.

Сложнось может возникнуть лишь с методом Ньютона — для него требуется уметь вычислять $F'(p)$. На практике применяют либо численное дифференцирование, либо решают дополнительное *уравнение в вариациях*, для нахождения $\frac{\partial y(x,p)}{\partial p}$.

```
from scipy.optimize import fsolve

def residual(p): # Невязка правого краевого условия
    print('Решаем задачу Коши при p =', *p)
    return shoot(*p) - 1

p_approx = 0.5
[p_true] = fsolve(residual, p_approx)
print('p_true =', p_true)
shoot_plot(p_true)
```

```
Решаем задачу Коши при p = 0.5
Решаем задачу Коши при p = 0.5
Решаем задачу Коши при p = 0.5
Решаем задачу Коши при p = 0.500000007451
Решаем задачу Коши при p = 0.439650556032
Решаем задачу Коши при p = 0.439968123456
Решаем задачу Коши при p = 0.439966522627
Решаем задачу Коши при p = 0.439966522586
p_true = 0.439966522586
```



Задачи на собственные значения

Рассмотрим задачу на собственные значения λ оператора Штурма-Лиувилля ($p(x), q(x), \rho(x)$ — заданые функции)

$$-\frac{d}{dx}\left[p(x)\frac{dy}{dx}\right] + q(x)y(x) = \lambda\rho(x)y(x), \qquad x \in [a,b]$$

$$\alpha_1 y(a) + \beta_1 y'(a) = 0$$

$$\alpha_2 y(b) + \beta_2 y'(b) = 0$$

Очевидно, что функция $y(x) = 0$ является решением этой задачи при любых λ. Однако лишь при некоторых λ кроме этого решения существует еще и нетривиальное $y(x) \not\equiv 0$.

Неединственность решения

Пусть $y_0(x)$ — некоторое нетривиальное решение однородной задачи

$$-\frac{d}{dx}\left[p(x)\frac{dy}{dx}\right] + q(x)y(x) = \lambda\rho(x)y(x), \qquad x \in [a,b]$$

$$\alpha_1 y(a) + \beta_1 y'(a) = 0$$

$$\alpha_2 y(b) + \beta_2 y'(b) = 0$$

Очевидно, что $Cy_0(x), C \neq 0$ также будет нетривиальным решением. Сравните с задачей для матрицы:

$$\mathbf{A}\mathbf{x} = \lambda\mathbf{x}$$

Собственный вектор $\mathbf{x}$ определен с точностью до умножения на число.

Необходимо поставить задачу так, чтобы при характеристических λ она имела ровно одно решение. Добавим дополнительное условие, нарушающее однородность: зададим в точке $x = a$ *еще одно* неоднородное краевое условие, например, $y(a) = 1$ или $y'(a) = 1$, линейно независимое с уже имеющимся в точке $x = a$ условием.

$$-\frac{d}{dx}\left[p(x)\frac{dy}{dx}\right] + q(x)y(x) = \lambda\rho(x)y(x), \qquad x \in [a,b]$$

$$\alpha_1 y(a) + \beta_1 y'(a) = 0$$

$$y'(a) = 1$$

$$\alpha_2 y(b) + \beta_2 y'(b) = 0$$

Задача превратилась в неоднородную краевую задачу втогоро порядка с *тремя* краевыми условиями и параметром λ. В общем случае она решений не имеет, но решение будет существовать и быть единственным при некоторых λ.

Временно закроем глаза на правое краевое условие:

$$-\frac{d}{dx}\left[p(x)\frac{dy}{dx}\right] + q(x)y(x) = \lambda\rho(x)y(x), \qquad x \in [a,b]$$

$$\alpha_1 y(a) + \beta_1 y'(a) = 0$$

$$y'(a) = 1$$

$$\alpha_2 y(b) + \beta_2 y'(b) = 0$$

Без него данная задача является полноценной задачей Коши, имеющей решение при любом значении параметра λ. Но только при некоторых λ решение этой задачи Коши дополнительно удовлетворяет еще и правым краевым условиям

$$\alpha_2 y(b) + \beta_2 y'(b) = 0.$$

Мы имеем полную аналогию с методом стрельбы, только параметром на этот раз является не недостающая производная на конце отрезка, а параметр уравнения λ.

Рассмотрим в качестве примера задачу определения минимального собственного числа $\lambda > 0$ и собственной функции $y(x)$:

$$-\frac{d}{dx}\left[(1+x^2)\frac{dy}{dx}\right] + x^2 y(x) = \lambda x y(x)$$

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

Добавим второе начальное условие $y(0) = 1$. Теперь данных достаточно, чтобы решить задачу Коши при заданном λ. Вместо того, чтобы сводить задачу к системе первого порядка (можно было бы и так), аппроксимируем эту задачу сразу.

Задача Коши

$$-\frac{d}{dx}\left[(1+x)\frac{dy}{dx}\right]+e^{-x}y(x)=\lambda y(x), \quad x\in[0,1]$$

$$y(0)+y'(0)=0,\ y(0)=1$$

может быть аппроксимирована со вторым порядком следующим образом:

$$-\frac{(1+x_{m+1/2})(u_{m+1}-u_m)-(1+x_{m-1/2})(u_m-u_{m-1})}{h^2}+(e^{-x_m}-\lambda)u_m=0,$$

$$m=1,2,\ldots,M-1$$

$$u_0=1$$

Второе начальное условие $y(0)+y'(0)=0$ с первым порядком аппроксимируется условием

$$u_0+\frac{u_1-u_0}{h}=0$$

Для получения аппроксимации второго порядка, введем $\phi(x) = 1 - \frac{x}{h}$ и проинтегрируем уравнение с этой функцией:

$$\int_0^h [(1+x)y']'\phi dx - \int_0^h ye^{-x}\phi dx + \lambda \int_0^h y\phi dx = 0$$

$$(1+x)y'\phi\Big|_0^h - \int_0^h (1+x)y'\phi' dx - \int_0^h ye^{-x}\phi dx + \lambda \int_0^h y\phi dx = 0$$

Значение $(1+x)y'\phi\Big|_0^h = (1+h)y'(h)\phi(h) - y'(0)\phi(0) = -y'(0)$

Таким образом,

$$y'(0) = -\int_0^h (1+x)y'\phi' dx - \int_0^h e^{-x}y\phi dx + \lambda \int_0^h y\phi dx \approx$$

$$\approx (1+h/2)\frac{y(h)-y(0)}{h} + (\lambda - 1)\frac{h}{2}y(0) + O(h^2)$$

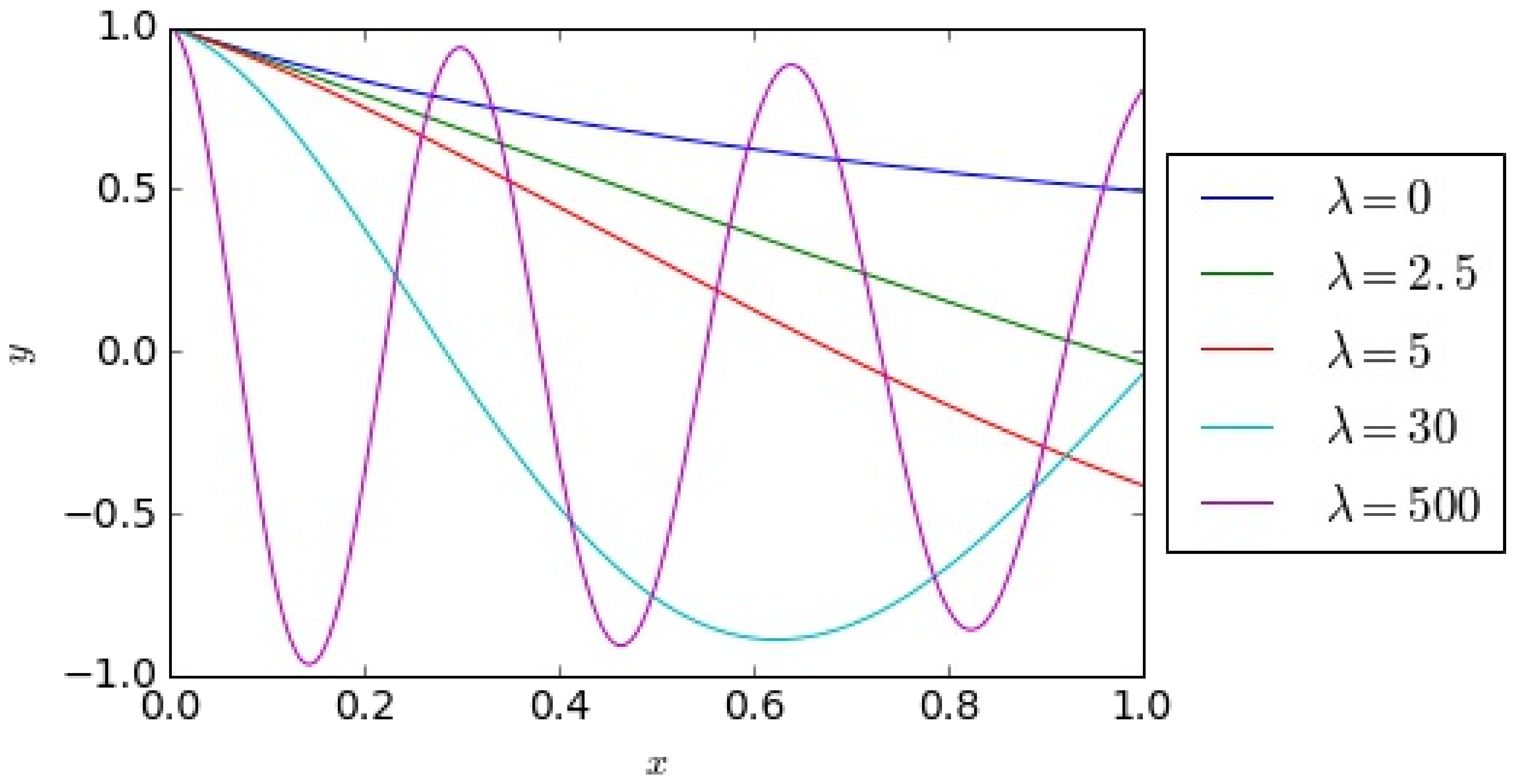
Краевое условие $y'(0) + y(0) = 0$ со вторым порядком аппроксимации принимает вид (синим отмечены отличия от простейшей аппроксимации первого порядка)

$$u_0 + (1 + h/2)\frac{u_1 - u_0}{h} + (\lambda - 1)\frac{hu_0}{2} = 0$$

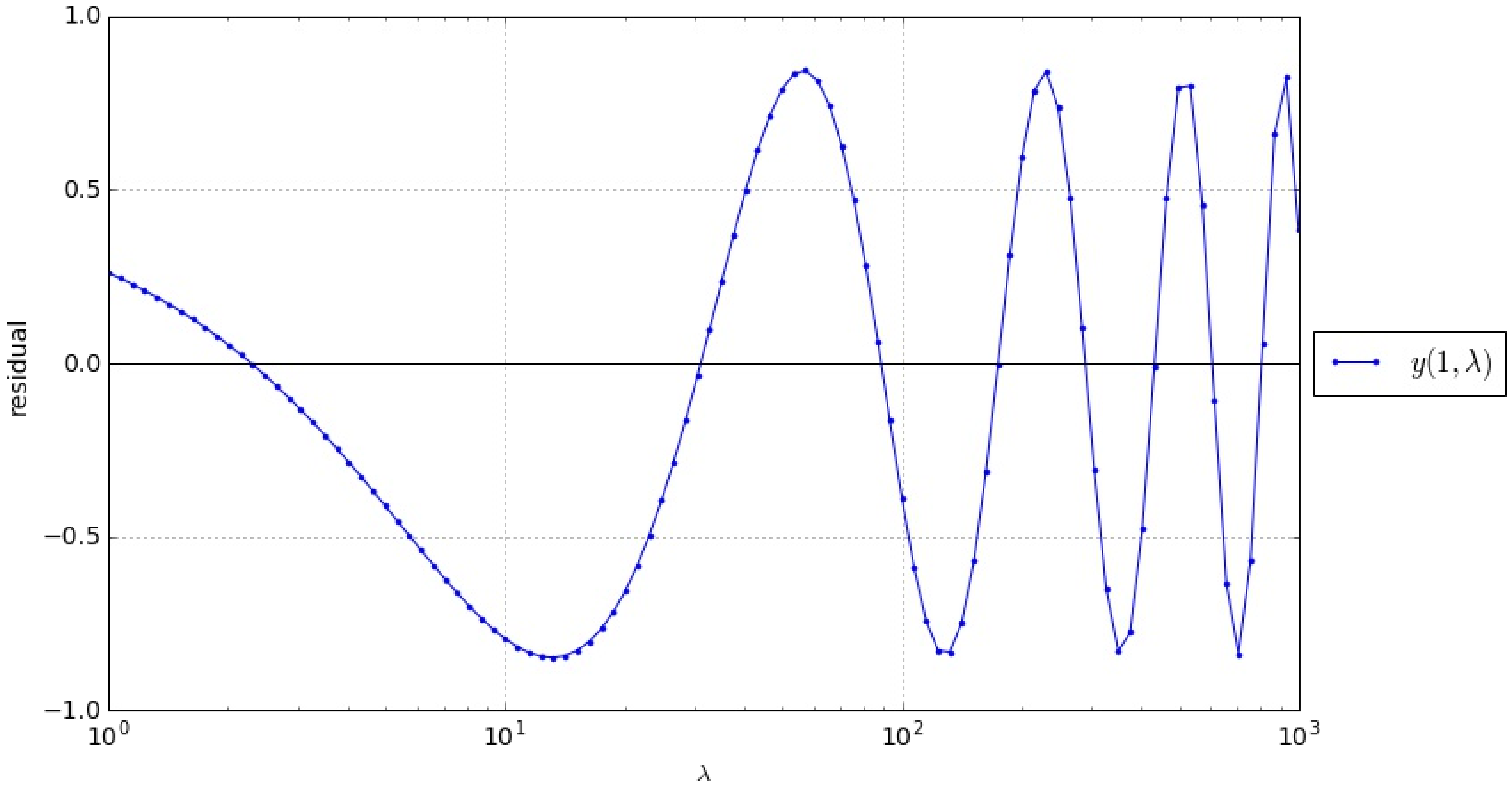
Отметим, что в последнем слагаемом можно заменить u_0 на u_1 без потери порядка аппроксимации, так как $y(h) = y(0) + O(h)$.

```python
def solve_cauchy(M, lam, order=2):
    x = np.linspace(0, 1, M+1); h=x[1]-x[0]
    u = np.empty_like(x)
    # Заполняем начальные условия
    u[0] = 1
    if order==1:
        fd = -u[0]
    else:
        fd = ((1-lam)*h*u[0]/2 - u[0]) / (1+h/2)
    u[1] = u[0] + h * fd
    for m in range(1, M):
        xm = m*h; xml = xm-h/2; xmp = xm+h/2
        rhs = (np.exp(-xm) - lam)*u[m]*h*h
        u[m+1] = u[m] + (rhs + (1+xml)*(u[m] - u[m-1])) / (1+xmp)
    return x,u

def residual(*args, **kwargs):
    _,u = solve_cauchy(*args, **kwargs)
    return u[-1]
```

```
X, U1 = solve_cauchy(200, 0, 2)
X, U2 = solve_cauchy(200, 2.5, 2)
X, U3 = solve_cauchy(200, 5, 2)
X, U4 = solve_cauchy(200, 30, 2)
X, U5 = solve_cauchy(200, 500, 2)
plt.plot(X, U1, label=r'$\lambda=0$')
plt.plot(X, U2, label=r'$\lambda=2.5$')
plt.plot(X, U3, label=r'$\lambda=5$')
plt.plot(X, U4, label=r'$\lambda=30$')
plt.plot(X, U5, label=r'$\lambda=500$')
plt.legend(loc='center left', bbox_to_anchor=(1., .5))
plt.xlabel('$x$'); plt.ylabel(r'$y$'); plt.show()
```



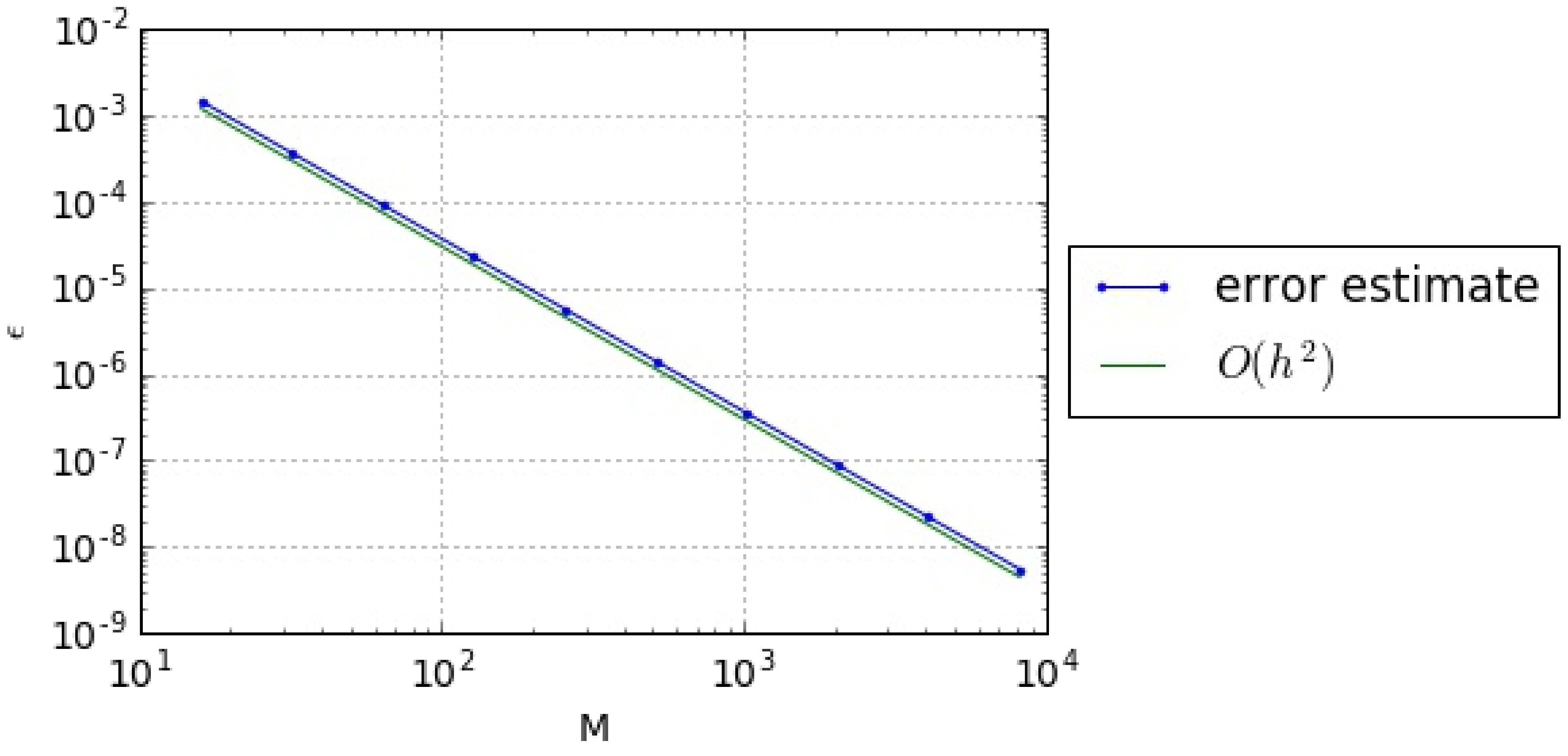
```
lambdas = np.logspace(0, 3, 100); val=[]
for lam in lambdas:
    val.append(residual(200, lam, order=2))
plt.figure(figsize=(12, 7))
plt.plot(lambdas, val, '.-', label=r'$y(1, \lambda)$')
plt.plot(lambdas, np.zeros_like(val), 'k')
plt.legend(loc='center left', bbox_to_anchor=(1.,.5))
plt.xlabel(r'$\lambda$'); plt.ylabel('residual')
plt.xscale('log'); plt.grid(); plt.show()
```



Вычислим первое собственное значение

Из графика следует, что $\lambda_1 \approx 2.5$. Уточним это значение численно решив уравнение $F(\lambda) = y(1, \lambda) = 0$

```
Ms = 2**np.arange(3, 14); Ls = []
for M in Ms:
    [val] = fsolve(lambda lam: residual(M, lam, order=2), Ls[-1] if len(Ls)>0 else 2.5, xtol=1e-12)
    Ls.append(val)
plt.loglog(Ms[1:], np.abs(np.diff(Ls)) / 3, '.-', label=r'error estimate')
plt.loglog(Ms[1:], 0.3 / Ms[1:]**2, '-', label=r'$O(h^2)$')
plt.xlabel('M'); plt.ylabel(r'$\epsilon$'); plt.grid()
plt.legend(loc='center left', bbox_to_anchor=(1.,.5)); plt.show()
```



Убедимся, что простейшая аппроксимация краевого условия дает только первый порядок сходимости для λ_1

```
Ms = 2**np.arange(3, 14); Ls = []
for M in Ms:
    [val] = fsolve(lambda lam: residual(M, lam, order=1), Ls[-1] if len(Ls)>0 else 2.5, xtol=1e-10)
    Ls.append(val)
plt.loglog(Ms[1:], np.abs(np.diff(Ls)), '.-', label=r'error estimate')
plt.loglog(Ms[1:], 0.5 / Ms[1:], '-', label=r'$O(h)$')
plt.xlabel('M'); plt.ylabel(r'$\epsilon$'); plt.grid()
plt.legend(loc='center left', bbox_to_anchor=(1.,.5)); plt.show()
```