Учреждение образования
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»
Кафедра информатики

Отчет по лабораторной работе №1
Решение краевых задач. Методы коллокаций, наименьших квадратов и Галеркина

Выполнил:
студент гр. 953505
Басенко К. А.

Руководитель:
доцент
Анисимов В. Я.

Минск 2021

Цель работы:

- изучить методы коллокаций, наименьших квадратов и Галеркина, составить алгоритмы методов и программы их реализаций, составить алгоритм решения краевых задач указанными методами, применимыми для организации вычислений на ПЭВМ;
- составить программу решения краевых задач по разработанным алгоритмам;
- выполнить тестовые примеры и проверить правильность работы программ.
- получить численное решение заданной краевой задачи.

Краткие теоретические сведения

Будем рассматривать дифференциальное уравнение второго порядка [3].

$$y'' + p(x)y' + q(x)y = f(x), \qquad (2.1)$$

где $p(x)$, $q(x)$, $f(x)$ – заданные непрерывные на отрезке $[a, b]$ функции.

Напомним, что задача Коши для уравнения (2.1) сводится к нахождению решения $y(x)$, удовлетворяющего начальным условиям:

$$\begin{cases} y(a) = A \\ y'(a) = A_1. \end{cases}$$

Краевой задачей называется задача нахождения решения $y(x)$, удовлетворяющего граничным условиям:

$$\begin{cases} y(a) = A, \\ y(b) = B. \end{cases}$$

Краевая задача отличается от задачи Коши непредсказуемостью. Ее решение может существовать, не существовать, быть единственным, может быть бесконечно много решений.

Часто вместо граничных условий используют обобщенные граничные условия:

$$\begin{cases}\alpha_1 y(a)+\beta_1 y'(a)=A,\\ \alpha_2 y(b)+\beta_2 y'(b)=B.\end{cases}$$

Граничные условия называются *однородными,* если $A = B = 0$.

Соответственно, краевая задача называется *однородной,* если у нее однородные граничные условия и правая часть уравнения $f(x) \equiv 0$. Следующая теорема имеет важное теоретическое значение.

Теорема. Краевая задача имеет решение, причем единственное тогда и только тогда, когда соответствующая ей однородная краевая имеет только нулевое решение (тривиальное решение однородной краевой задачи).

Способы решения краевой задачи

Поскольку достаточно хороших аналитических методов нет, то для отыскания решения краевой задачи используются приближенные методы. Приближенное решение строят в виде линейной комбинации функций [4]:

$$y_n(x) = \varphi_0(x) + a_1\varphi_1(x) + ... + a_n\varphi_n(x), \qquad (2.2)$$

где $\varphi_0(x)$ удовлетворяет граничному условию, а функции $\varphi_1(x),...,\varphi_n(x)$ – линейно независимы на [*a, b*] и удовлетворяют однородным граничным условиям.

Такая система дважды непрерывно дифференцируемых функций $\varphi_0(x),\varphi_1(x),...,\varphi_n(x)$ называется *базисной системой*. Задача сводится к выбору коэффициентов $a_1,...,a_n$ таких, чтобы функция $y_n(x)$ удовлетворяла граничному условию и была в некотором смысле близкой к точному решению.

Подставим приближенное решение (2.2) в уравнение (2.1). Полученное выражение

$$\psi(x,a_1,...,a_n) = y_n''(x) + p(x)y_n'(x) + q(x)y_n(x) - f(x) \qquad (2.3)$$

называют невязкой. Очевидно, что, если бы $\psi(x,a_1,...,a_n) \equiv 0$, то $y_n(x)$ было бы точным решением. К сожалению, так бывает очень редко. Следовательно, необходимо выбрать коэффициенты таким образом, чтобы невязка была в некотором смысле минимальной.

Метод коллокаций

На отрезке $[a,\ b]$ выбираются точки $x_1,...,x_m \in [a,b]$ $(n \geq m)$, которые называются точками коллокации. Точки коллокации последовательно подставляются в невязку. Считая, что невязка должна быть равна нулю в точках коллокации, в итоге получаем систему уравнений для определения коэффициентов $a_1,...,a_n$.

$$\begin{cases} \psi(x_1, a_1, ..., a_n) = 0, \\ ... \\ \psi(x_m, a_1, ..., a_n) = 0. \end{cases}$$

Обычно $m = n$. Получается система из n линейных уравнений с n неизвестными (коэффициентами $a_1,...,a_n$):

$$\begin{cases} \psi(x_1, a_1, ..., a_n) = 0, \\ ... \\ \psi(x_n, a_1, ..., a_n) = 0. \end{cases}$$

Решая эту систему, найдем приближенное решение $y_n(x)$. Для повышения точности расширяем систему базисных функций. В значительной степени успех в применении метода зависит от удачного выбора базисной системы.

ЗАДАНИЕ К ЛАБОРАТОРНОЙ РАБОТЕ №11

Методами коллокаций, галеркина, интегральным и дискретным методами наименьших квадратов и получить численное решение краевой задачи:

$$ay'' + (1 + bx^2)y = -1, \qquad -1 \le x \le 1.$$

Исходные данные:

$$a = \sin(k), b = \cos(k),$$

где k номер варианта. Базисную систему выбрать в виде:

$$\varphi_0 = 0,$$
$$\varphi_i(x) = x^i(1 - x^2), \qquad i = 1, 2, \ldots$$

Граничные условия:

$$y(-1) = 0,$$
$$y(1) = 0.$$

Программная реализация:

Записываем условие

```
def phi(i, x):
    return np.power(x, i)*(1-x*x)

def ddphi(i, x):
    if i == 0:
        return 0
    elif i == 1:
        return -2
    elif i == 2:
        return -6*x
    return -(i + 1) * np.power(x, i - 2) * (1 + (i + 2)*x*x)

def q(x) -> float:
    return 1+b*x*x

def f(x):
    return 1

def y(x):
    res = phi(0, x)
    for i in range(n):
        res += ais[i] * phi(i + 1, x)
    return res
```

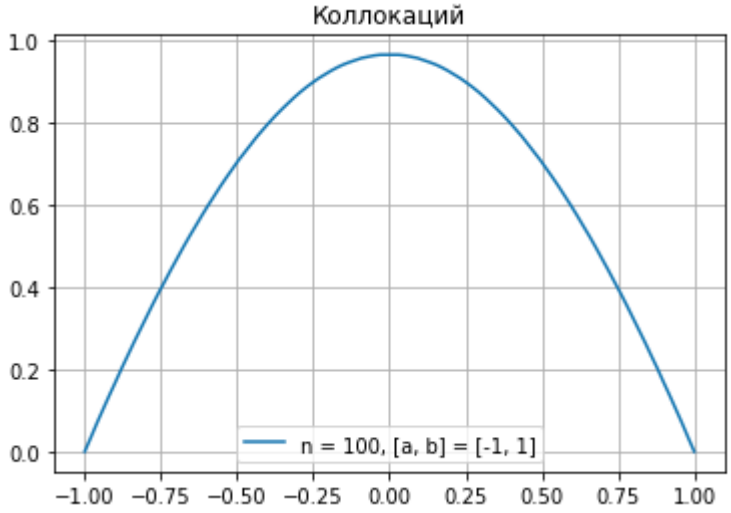
Дописываем:

```
def k(i, x):
    return a * ddphi(i, x) + q(x) * phi(i, x)

def c(x):
    return f(x) - a * ddphi(0, x) - q(x) * phi(0, x)
```

```
n = 100
xs = [round(x, 3) for x in np.linspace(A, B, n)]

syst = []
vec = []

for x in xs:
    r = []
    for i in range(n):
        r.append(k(i+1, x))
    syst.append(r)
    vec.append(c(x))

ais = np.linalg.solve(syst, vec)
ys = [round(y(x), 3) for x in xs]
ais = [round(a, 3) for a in ais]

grid_func = list(zip(xs, ys))
print("first 10 of ai coeffs:\n", list(ais[:10]))
print()
print("some points of grid func:\n(x, y(x))\n", grid_func[:10], "...\n...", grid_func[-10:]
    if len(grid_func) > 10
    else grid_func)

plt.plot(xs, ys, label=f"n = {n}, [a, b] = [{A}, {B}]")
plt.title("Коллокаций")
plt.legend()
plt.grid()
plt.show()
```

Фиксируем:

```
first 10 of ai coeffs:
 [0.964, 0.0, -0.116, 0.0, 0.02, 0.0, -0.001, 0.0, 0.0, -0.0]

some points of grid func:
(x, y(x))
 [(-1.0, 0.0), (-0.98, 0.034), (-0.96, 0.069), (-0.939, 0.104), (-0.919, 0.137),
(-0.899, 0.169), (-0.879, 0.202), (-0.859, 0.233), (-0.838, 0.266), (-0.818, 0.296)] ...
... [(0.818, 0.296), (0.838, 0.266), (0.859, 0.233), (0.879, 0.202), (0.899, 0.169),
(0.919, 0.137), (0.939, 0.104), (0.96, 0.069), (0.98, 0.034), (1.0, 0.0)]
```

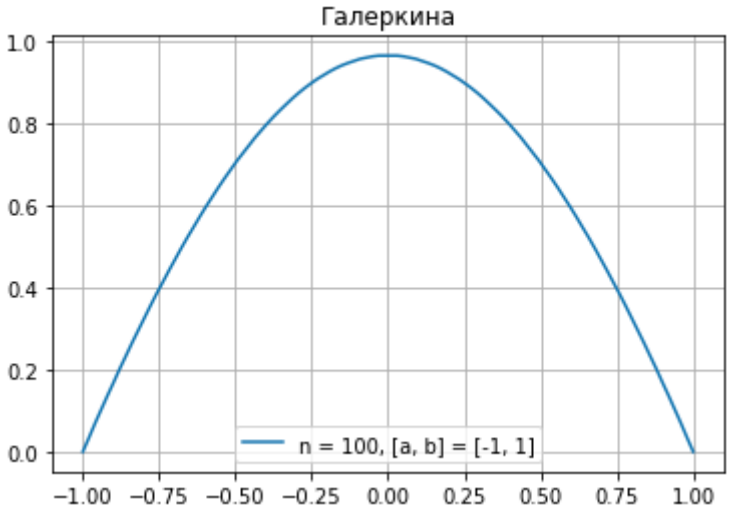


Дописываем:

```
def k(i, j):
    res = a * integrate.quad(lambda x: phi(i, x) * ddphi(j, x), A, B)[0] + \
          integrate.quad(lambda x: q(x) * phi(i, x) * phi(j, x), A, B)[0]
    return res

def c(i):
    res = -(integrate.quad(lambda x: phi(i, x) * ddphi(0, x), A, B)[0] + \
          integrate.quad(lambda x: q(x) * phi(i, x) * phi(0, x), A, B)[0] + \
          integrate.quad(lambda x: phi(i, x), A, B)[0])
    return res

for i in range(1, n + 1):
    r = []
    for j in range(1, n + 1):
        r.append(k(i, j))
    syst.append(r)
    vec.append(c(i))

ais = np.linalg.solve(syst, vec)

ais = np.linalg.solve(syst, vec)
ys = [round(y(x), 3) for x in xs]
ais = [round(a, 3) for a in ais]

grid_func = list(zip(xs, ys))
print("first 10 of ai coeffs:\n", list(ais[:10]))
print()
print("some points of grid func:\n(x, y(x))\n", grid_func[:10], "...\n...", grid_func[-10:]
    if len(grid_func) > 10
    else grid_func)

plt.plot(xs, ys, label=f"n = {n}, [a, b] = [{A}, {B}]")
plt.title("Галеркина")
plt.legend()
plt.grid()
plt.show()
```

Фиксируем:

```
first 10 of ai coeffs:
 [0.964, -0.0, -0.116, -0.0, 0.02, -0.0, -0.001, -0.0, -0.0, -0.0]

some points of grid func:
(x, y(x))
 [(-1.0, 0.0), (-0.98, 0.034), (-0.96, 0.069), (-0.939, 0.104), (-0.919, 0.137),
(-0.899, 0.169), (-0.879, 0.202), (-0.859, 0.233), (-0.838, 0.266), (-0.818, 0.296)] ...
... [(0.818, 0.296), (0.838, 0.266), (0.859, 0.233), (0.879, 0.202), (0.899, 0.169),
(0.919, 0.137), (0.939, 0.104), (0.96, 0.069), (0.98, 0.034), (1.0, 0.0)]
```



Дописываем:

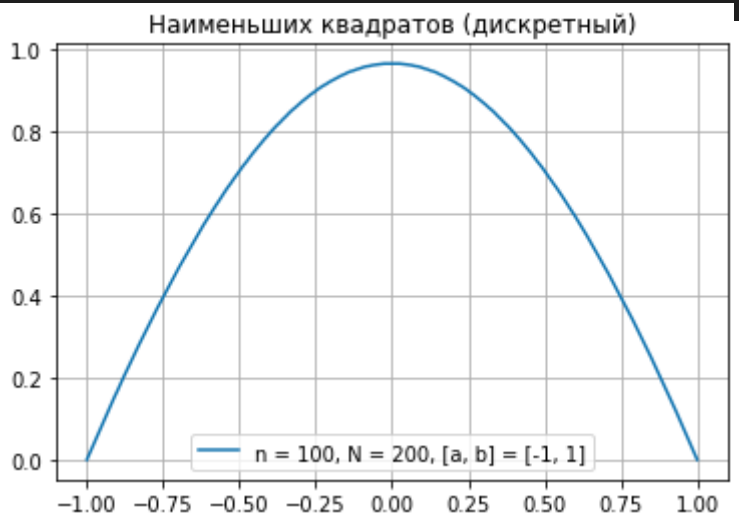
```
N = 2*n

xs = [round(x, 3) for x in np.linspace(A, B, N)]
```

```
def k(i, j):
    res = 0
    for x in xs:
        res += (a * ddphi(i, x) + q(x)*phi(i, x)) * \
               (a * ddphi(j, x) + q(x) * phi(j, x))
    res *= 2
    return res

def c(i):
    res = 0
    for x in xs:
        res += (a * ddphi(i, x) + q(x)*phi(i, x)) * \
               (a * ddphi(0, x) + q(x) * phi(0, x) + 1)
    res *= -2
    return res
```

```
syst = []
vec = []

for i in range(1, n + 1):
    r = []
    for j in range(1, n + 1):
        r.append(k(i, j))
    syst.append(r)
    vec.append(c(i))

ais = np.linalg.solve(syst, vec)

ais = np.linalg.solve(syst, vec)
ys = [round(y(x), 3) for x in xs]
ais = [round(a, 3) for a in ais]

grid_func = list(zip(xs, ys))
print("first 10 of ai coeffs:\n", list(ais[:10]))
print()
print("some points of grid func:\n(x, y(x))\n", grid_func[:10], "...\n...", grid_func[-10:]
    if len(grid_func) > 10
    else grid_func)

plt.plot(xs, ys, label=f"n = {n}, N = {N}, [a, b] = [{A}, {B}]")
plt.title("Наименьших квадратов (дискретный)")
plt.legend()
plt.grid()
plt.show()
```

Фиксируем:

```
first 10 of ai coeffs:
 [0.964, -0.0, -0.116, 0.0, 0.02, -0.0, -0.001, 0.0, 0.0, -0.003]

some points of grid func:
(x, y(x))
 [(-1.0, 0.0), (-0.99, 0.017), (-0.98, 0.034), (-0.97, 0.052), (-0.96, 0.069), (-0.95,
0.085), (-0.94, 0.102), (-0.93, 0.119), (-0.92, 0.135), (-0.91, 0.152)] ...
... [(0.91, 0.152), (0.92, 0.135), (0.93, 0.119), (0.94, 0.102), (0.95, 0.085), (0.96,
0.069), (0.97, 0.052), (0.98, 0.034), (0.99, 0.017), (1.0, 0.0)]
```



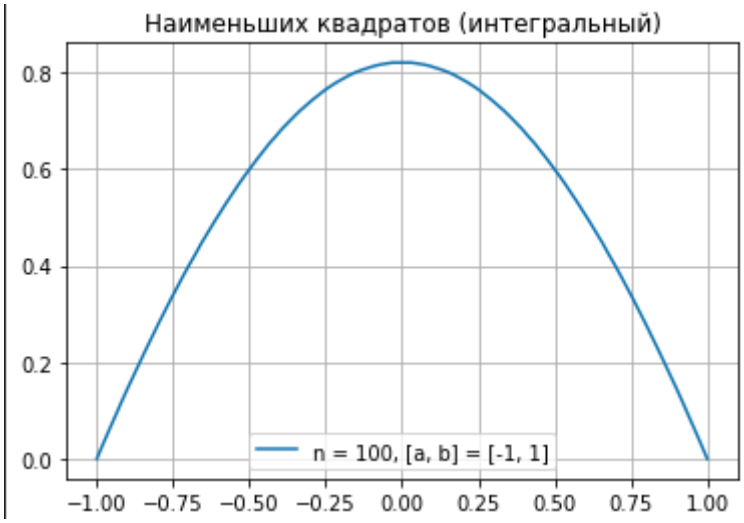
Дописываем:

```
def k(i, j):
    res = integrate.quad(
        lambda x: \
            (ddphi(j, x) + q(x) * phi(j, x)) * \
            (ddphi(i, x) + q(x) * phi(i, x)),
        A, B)[0]
    return res

def c(i):
    res = integrate.quad(
        lambda x: f(x) * ( ddphi(i, x) + q(x)*phi(i, x)),
        A, B)[0]
    return res
```

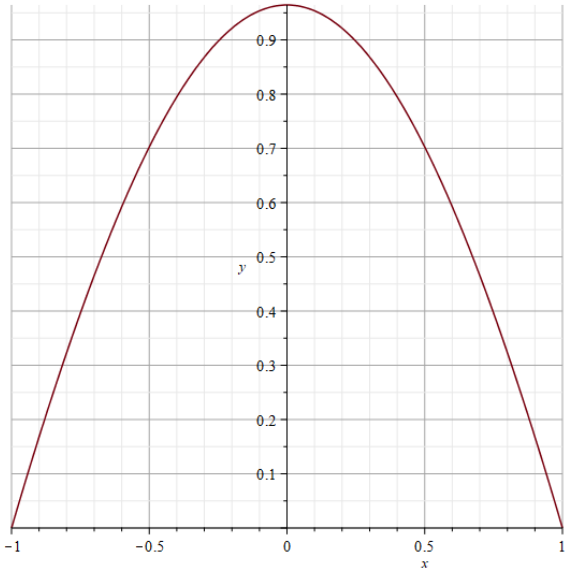
```
syst = []
vec = []

for i in range(1, n+1):
    r = []
    for j in range(1, n+1):
        r.append(k(i, j))
    syst.append(r)
    vec.append(c(i))

ais = np.linalg.solve(syst, vec)

ais = np.linalg.solve(syst, vec)
ys = [round(y(x), 3) for x in xs]
ais = [round(a, 3) for a in ais]

grid_func = list(zip(xs, ys))
print("first 10 of ai coeffs:\n", list(ais[:10]))
print()
print("some points of grid func:\n(x, y(x))\n", grid_func[:10], "...\n...", grid_func[-10:]
    if len(grid_func) > 10
    else grid_func)

plt.plot(xs, ys, label=f"n = {n}, [a, b] = [{A}, {B}]")
plt.title("Наименьших квадратов (интегральный)")
plt.legend()
plt.grid()
plt.show()
```

Фиксируем:

```
first 10 of ai coeffs:
 [0.821, 0.0, -0.089, -0.0, 0.015, -0.0, -0.001, -0.0, 0.0, -0.0]

some points of grid func:
(x, y(x))
 [(-1.0, 0.0), (-0.98, 0.03), (-0.96, 0.059), (-0.939, 0.089), (-0.919, 0.118), (-0.899,
0.146), (-0.879, 0.173), (-0.859, 0.2), (-0.838, 0.228), (-0.818, 0.254)] ...
... [(0.818, 0.254), (0.838, 0.228), (0.859, 0.2), (0.879, 0.173), (0.899, 0.146),
(0.919, 0.118), (0.939, 0.089), (0.96, 0.059), (0.98, 0.03), (1.0, 0.0)]
```



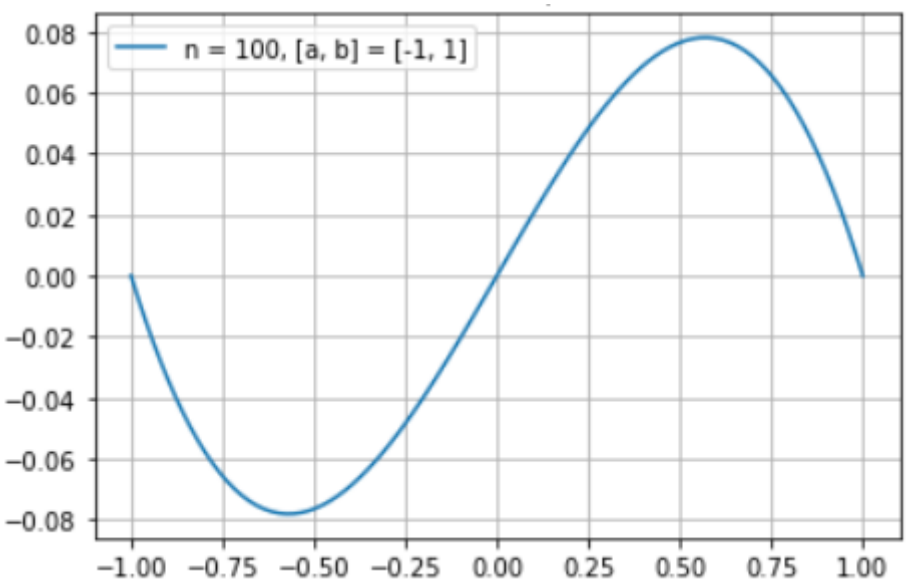
$$a := \sin(2) :$$
$$b := \cos(2) :$$
$$r := dsolve\left(\left\{y''(x) + \frac{1 + b \cdot x^2}{a} \cdot y(x) = -\frac{1}{a}, y(-1) = 0, y(1) = 0\right\}, numeric\right) :$$
$$odeplot(r, gridlines)$$



Тестовые примеры:

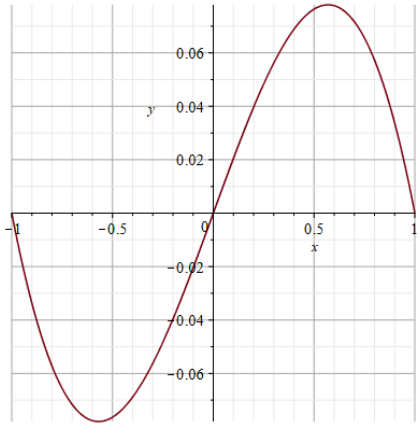
#1

```
def q(x):
    return (1+b*x*x)

def f(x):
    return -x
```



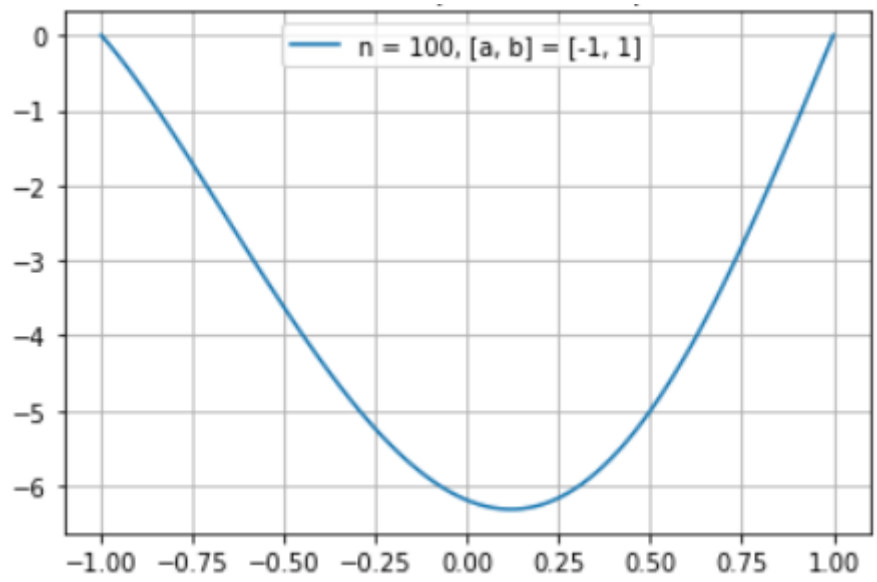
$$r := dsolve\left(\left\{ y''(x) + \frac{1 + b \cdot x^2}{a} \cdot y(x) = -\frac{x}{a}, y(-1) = 0, y(1) = 0 \right\}, numeric\right) :$$

$$odeplot(r, gridlines)$$

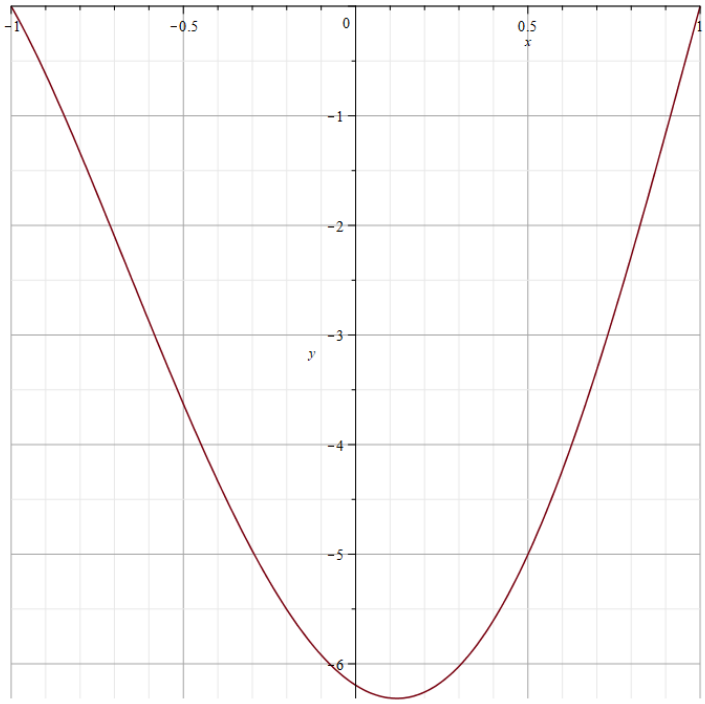


#2

```
def q(x):
    return np.sqrt(x + np.exp(x+2))

def f(x):
    return -np.exp(x*x)/(1 + np.sin(x))
```



$$r := dsolve\left(\left\{y''(x) + \mathrm{sqrt}(x + \exp(x+2))\cdot y(x) = -\frac{\exp(x\cdot x)}{1+\sin(x)}, y(-1) = 0, y(1) = 0\right\}, numeric\right):$$
$$odeplot(r, gridlines)$$



Заключение: все цели достигнуты