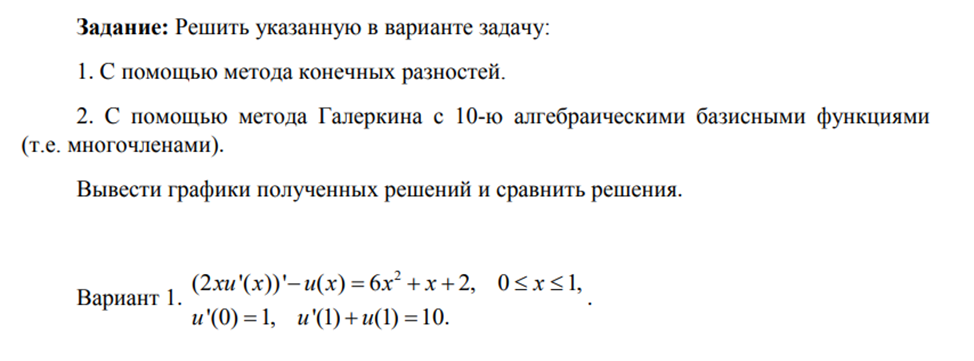
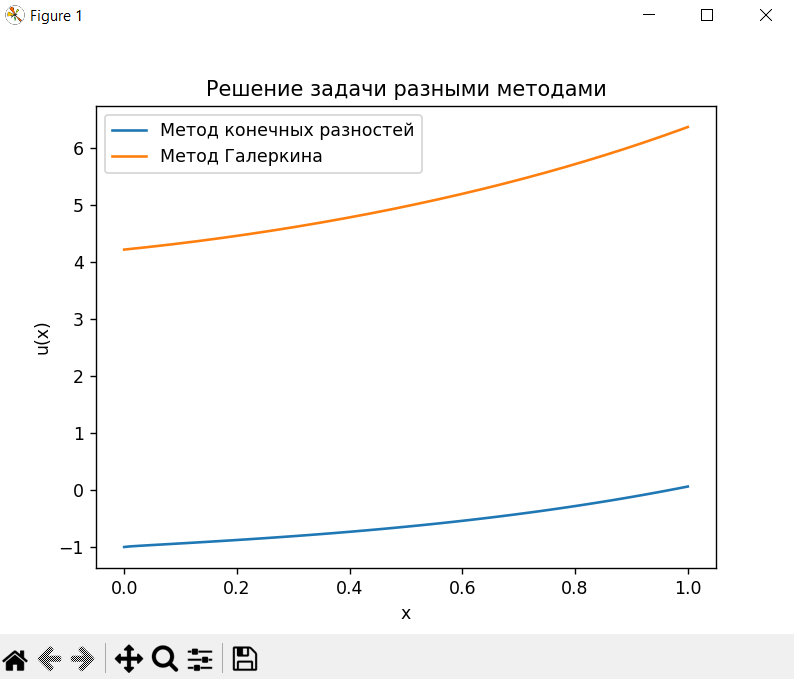
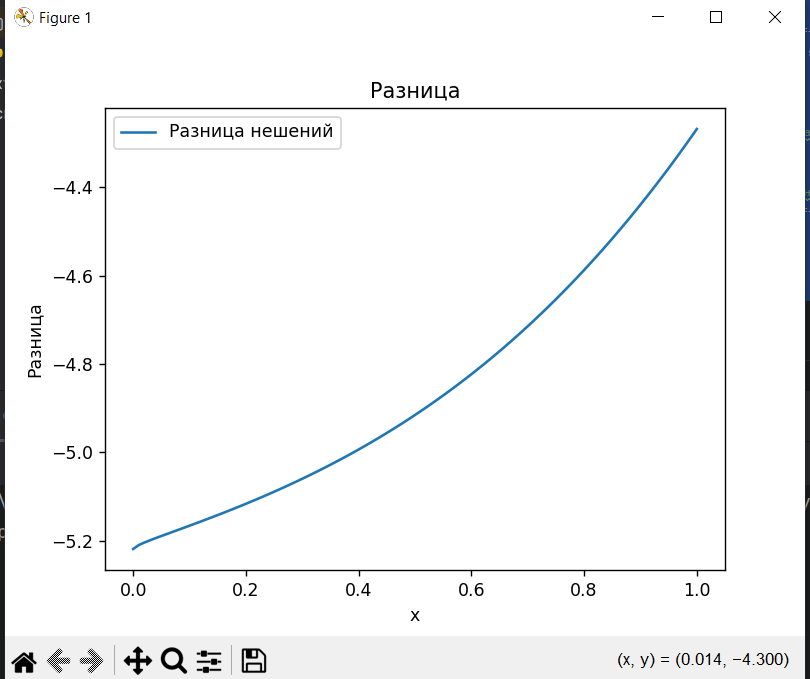
**Постановка задачи:**

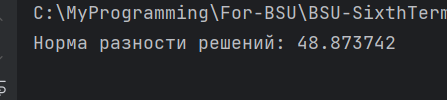
**Программная реализация:**

import numpy as np  
import matplotlib.pyplot as plt  
from scipy.integrate import quad  
from scipy.interpolate import interp1d  
  
  
# Метод конечных разностей  
def finite\_difference\_method(N):  
 x = np.linspace(0, 1, N + 1)  
 h = 1.0 / N  
  
 A = np.zeros((N + 1, N + 1))  
 b = np.zeros(N + 1)  
  
 # Внутренние узлы  
 for i in range(1, N):  
 xi = x[i]  
 A[i, i - 1] = ((2 \* xi) / (h \*\* 2)) - (1/h)  
 A[i, i] = ((-4 \* xi) / (h \*\* 2)) - 1  
 A[i, i + 1] = ((2 \* xi) / (h \*\* 2)) + (1/h)  
 b[i] = 6 \* (xi \*\* 2) + xi + 2  
  
 # Граничные условия  
 A[0, 0] = -1 / h  
 A[0, 1] = 1 / h  
 b[0] = 1  
  
 A[N, N - 1] = 1 / (2\*h)  
 A[N, N] = 3 / (2\*h)  
 b[N] = 10  
  
 # Решение системы  
 u = np.linalg.solve(A, b)  
  
 return x, u  
  
  
# Метод Галеркина  
def phi(i, x):  
 if i==0:  
 return x+8  
 else:  
 return -i-2+(x\*\*(i+1))  
 # return x\*\*i  
  
def dphi(i, x):  
 if i == 0:  
 return 1  
 else:  
 return (i+1)\*(x\*\*(i))  
 # if i==0:  
 # return 0  
 # else:  
 # return i \* x\*\*(i-1)  
  
def dphi2(i, x):  
 if i == 0:  
 return 0  
 else:  
 return i\*(i+1)\*(x\*\*(i-1))  
 # return i \* (i-1) \* x\*\*(i-2)  
  
def first(i,j):  
 integrative\_first = lambda x: dphi(j,x)\*phi(i,x)  
 return quad(integrative\_first, 0,1)[0]  
  
def second(i,j):  
 integrative\_second = lambda x: dphi2(j,x)\*phi(i,x)  
 return quad(integrative\_second, 0, 1)[0]  
  
def third(i,j):  
 integrative\_third = lambda x: phi(j,x)\*phi(i,x)  
 return quad(integrative\_third, 0, 1)[0]  
  
def fourth(i):  
 integrative\_fourth = lambda x: ((6\*(x\*\*2)) + x + 2)\*phi(i,x)  
 return quad(integrative\_fourth, 0,1)[0]  
  
def galerkin\_method(n\_basis):  
 A = np.zeros((n\_basis, n\_basis))  
 B = np.zeros(n\_basis)  
  
 for i in range(n\_basis):  
 for j in range(n\_basis):  
 A[i, j] = 2\*first(i,j) + 2\*second(i,j) - third(i,j)  
 B[i] = fourth(i)  
  
 # Граничные условия  
 A[0, :] = [dphi(i, 0) for i in range(0, n\_basis)]  
 B[0] = 1  
  
 A[-1, :] += [dphi(i, 1) + phi(i, 1) for i in range(0, n\_basis)]  
 B[-1] += 10  
  
 c = np.linalg.solve(A, B)  
  
 def u(x):  
 return sum(c[i] \* phi(i, x) for i in range(0, n\_basis))  
  
 x\_vals = np.linspace(0, 1, 1000)  
 u\_vals = [u(xi) for xi in x\_vals]  
 return x\_vals, u\_vals  
  
  
# Параметры  
N = 100  
M = 10  
  
# Решение методами  
x\_finite\_difference, solution\_finite\_difference = finite\_difference\_method(N)  
x\_galerkin, solution\_galerkin = galerkin\_method(M)  
  
  
res\_diff = solution\_finite\_difference - solution\_galerkin[:len(solution\_finite\_difference)]  
norm\_diff = np.linalg.norm(res\_diff)  
  
# Графики  
plt.plot(x\_finite\_difference, solution\_finite\_difference, label='Метод конечных разностей')  
plt.plot(x\_galerkin, solution\_galerkin, label='Метод Галеркина')  
plt.xlabel('x')  
plt.ylabel('u(x)')  
plt.legend()  
plt.title('Решение задачи разными методами')  
plt.show()  
  
print(f'Норма разности решений: {norm\_diff:.6f}')  
plt.plot(x\_finite\_difference, res\_diff, label='Разница нешений')  
plt.xlabel('x')  
plt.ylabel('Разница')  
plt.legend()  
plt.title('Разница')  
plt.show()

**Полученные результаты:**

****

****

****

Значение нормы разности решений между методом конечных разностей и методом Галеркина составляет приблизительно 48.87. Это означает, что решения, полученные этими двумя методами, достаточно различны на всем интервале.

Вероятно, это связано с тем, что метод конечных разностей и метод Галеркина являются разными численными методами, и каждый из них имеет свои ограничения и приближения при решении дифференциальных уравнений. Различия в результатах могут быть вызваны различными аппроксимациями и численными алгоритмами, используемыми в этих методах.