Решение интегрального уравнения I рода методом механических квадратур

Постановка задачи

Решить данное интергальное уравнение I рода методом коллокаций с использованием полиномов Чебышева.

$$\int_a^b K(x,t)u(t)dt = f(x) \tag{1}$$

Условие задачи:

$$\begin{aligned} &K(x,t) = \frac{1}{(3+x+t} \\ &f(x) = 2(\sqrt{2}-1) - 2\sqrt{x+2}\left(\arctan\sqrt{x+2} - \arctan\sqrt{\frac{x}{2}+1}\right) \\ &[a,b] = [0,1] \end{aligned} \tag{2}$$

Вывод формул

Необходимо найти функцию u, минимизирующую регуляризатор $\alpha\|u\|^2 + \|Ku - f\|^2$, где α - варьируемый параметр.
В методе коллокаций приближенное решение ищется в виде линейной комбинации по заданной системе координатных функций:

$$\tilde{u}(x) = \sum_{k=0}^{n} c_k \varphi_k(x) \tag{3}$$

Получаем систему уравнений:

$$\alpha \sum_{k=0}^{n} c_k \varphi_k(x_i) + \sum_{l=0}^{n} \int_b^a c_l K(x_i,t)\varphi_k(t)dt = f(x_i) \tag{4}$$

В качестве координатных функций используем полиномы Чебышева:

$$T_k(x) = \cos(k \arccos(x)) \tag{5}$$

При уменьшении параметра α решение приближается к точному до некоторого предела, связанного с неустойчивостью, возникающей из-за ошибок округления.

Текст программы

```
eps = 0.000000001; m = 30
alpha = float(raw_input('alpha = '))
n = int(raw_input('n=')); a = 0.0; b = 1.0; h = (b - a) / n
ff = []; fi = []; AA = zeros((n,n)); u = []; xx=[]
def poli_lezh(x,n):
    p0=1; p1=x
    for i in range(2,n+1):
        pi = 1.0/float(i) * ( (2.0*i - 1.0)*x*p1 - (i - 1.0)*p0 )
        p0 = p1; p1 = pi
    pp = n / (1 - x**2.0) * ( p0 - x*p1)
    return pi, pp
def roots_lezh(n):
    t = []
    for i in range(1,n+1):
        tk = -2
        t0 = np.cos (np.pi * (4.0*i - 1.0) / (4.0 * n + 2) )
        while abs(tk - t0) > eps:
            tk = t0 - poli_lezh(t0,n)[0] / poli_lezh(t0,n)[1]
            t0 = tk
        t.append(tk)
    return t
def weights_lezh(n):
    a = []; r = roots_lezh(n)
    for i in range(n):
        a.append ( 2.0 / (1 - r[i]**2.0) / poli_lezh(r[i],n)[1]**2 )
    return a
roots = roots_lezh(n); weights = weights_lezh(n)
for i in range(n):
    roots[i] = 0.5*roots[i] + 0.5
    weights[i] = weights[i]/2.0
def poli_cheb(x,n):
    t0=1.0; t1=2.0*x - 1.0
    for i in range(2,n):
        x = 2.0 * x - 1.0
        ti = 2.0 * x * t1 - t0
        t0 = t1; t1 = ti
    return t1
def Kernel(x,t):
    return 1.0 / (3.0 + x + t)
def func(x):
    return 2.0 * (np.sqrt(2) - 1.0 ) - 2.0 * np. sqrt (x + 2.0) * ( np.arctan (np
        .sqrt (x + 2.0) ) - np.arctan (np.sqrt (x/2.0 + 1.0) ) )
def int_mmk(f1,f2,n):
    return sum ( [ weights[i] * f1 * f2 for i in range(n) ] )
for i in range(n):
    xi = a + h*i; xx.append(xi)
    for k in range(n):
        K2 = int_mmk ( Kernel(roots[k],xi), Kernel(roots[k],roots[k]), n)
        AA[i,k] = int_mmk (K2, poli_cheb(roots[k],k), n )
    AA[i,i] = AA[i,i] + alpha*poli_cheb(xi,i)
    ff.append(int_mmk(Kernel(roots[k],xi),func(roots[k]),n))
c = solve(AA, ff)
for i in range(n):
    u.append ( sum([c[k]*poli_cheb(xx[i],k) for k in range(n)]) )
```

:

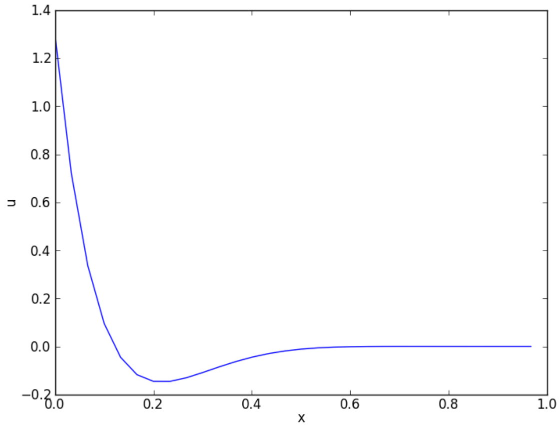

Рис. 1: α=1, n=10

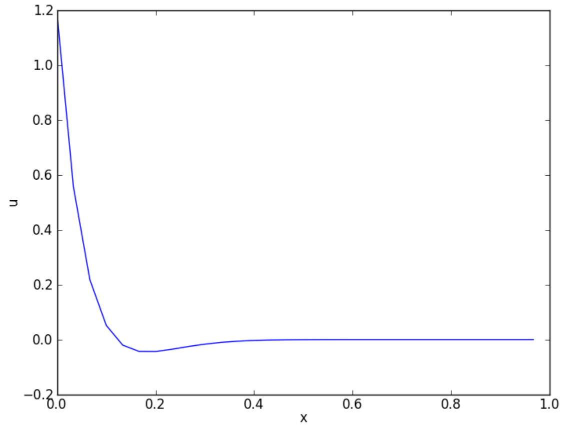

Рис. 2: α=1, n=15

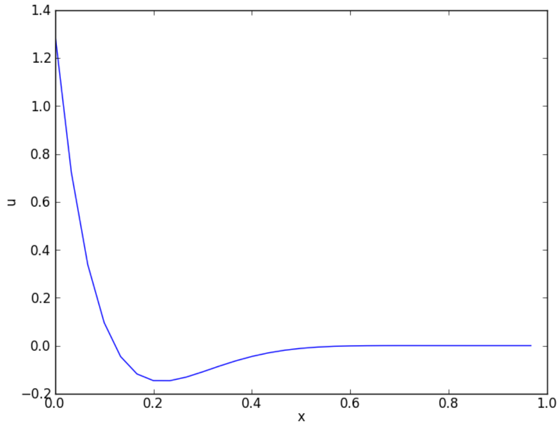

Рис. 3: α=0.0001, n=10

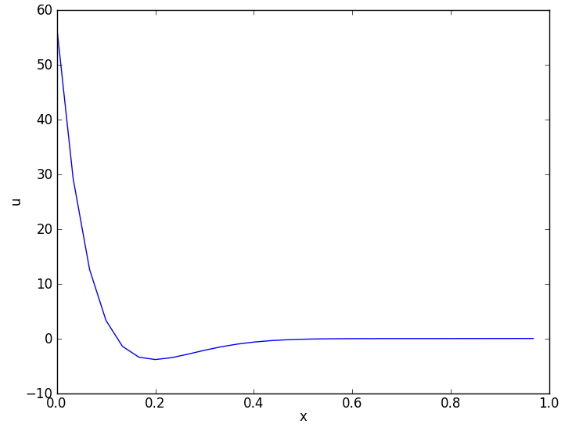

Рис. 4: α=0.0001, n=15

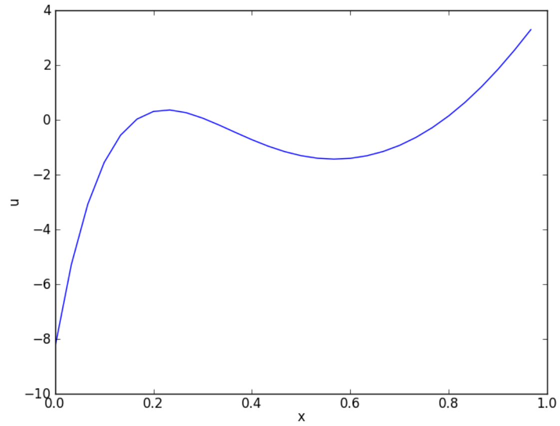

Рис. 5: α=10^{-8}, n=10

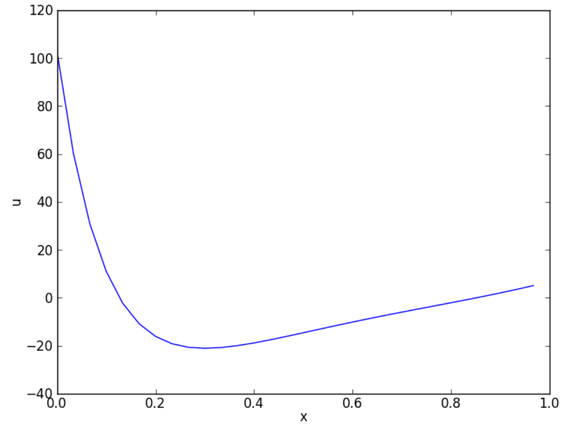

Рис. 6: α=10^{-8}, n=15

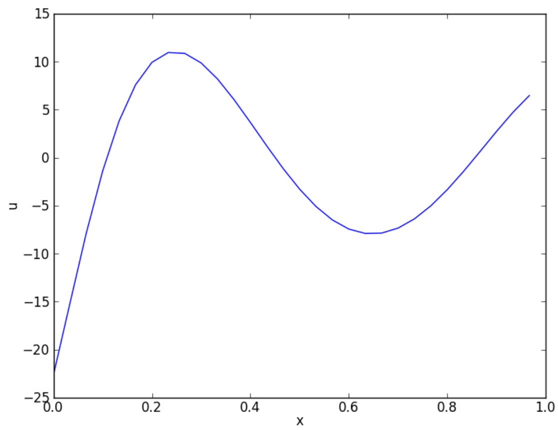

Рис. 7: α=10^{-12}, n=15