Решение интегрального уравнения первого рода методом коллокаций

***

Курдоякова Марина

Задание: Дано уравнение первого рода

$$\int_a^b K(x,t)u(t)dt = f(x)$$

Требуется привести уравнение к интегральному уравнению второго рода (или к интегро-дифференциальному виду) и решить его одним из предложенных методов. В моем элучае это метод коллокаций.

Даны следующие данные (5 вариант):

$$K = \frac{1}{3/2 + x + t)^2}$$

$$f = \frac{1}{1+x}\ln\frac{3(2x+3)}{2x+5} - \frac{\ln 2}{x+3/2} - \frac{\ln 3/2}{x+5/2}$$

Алгоритм решения задачи: Дано уравнение первого рода

$$\int_a^b K(x,t)u(t)dt = f(x)$$

Методом слабой регуляризации приходим к уравнению второго рода

$$\alpha u(x) + \int_a^b T(x,t)u(t)dt = \overline{f}(x)$$

где

$$T(x,t) = \int_a^b K(y,x)K(y,t)dy, \quad \overline{f}(x) = \int_a^b K(y,x)f(y)dy$$

Представим u в виде

$$u(x) = \sum_{k=1}^{n} C_k\phi_k(x)$$

и возьмем в качестве $\phi_k(x_i) = \cos(n\arccos(x_i)$. Теперь, после подстановки получается система уравнений с неизвестными C_k. После их нахождения подставляя, находим наше решение $u(x)$.

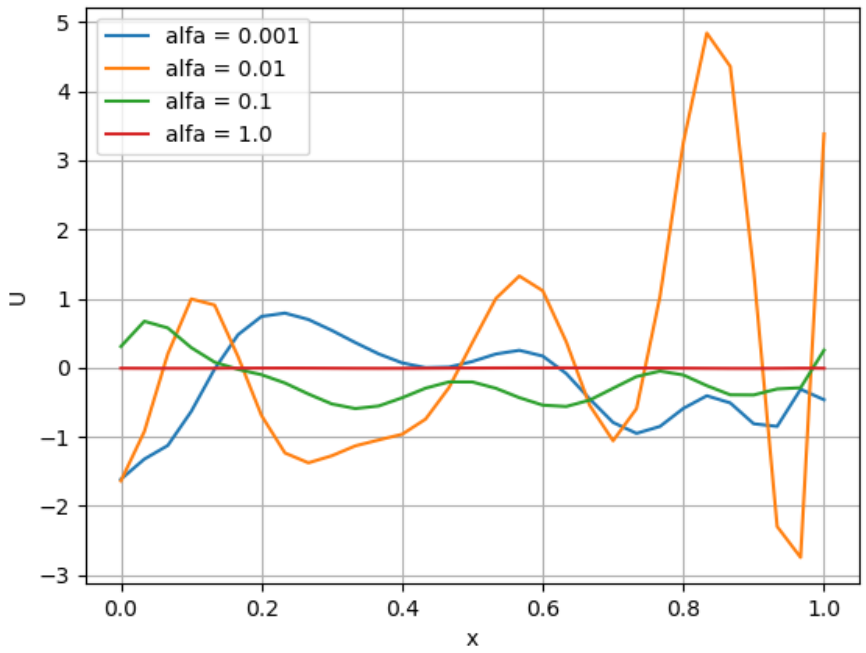
Код программы:

```
import numpy as np
from math import sqrt, sin, pi, cos, log, exp, acos, log
from scipy.integrate import dblquad, quad
import matplotlib.pyplot as plt

aa, bb = [0,1]
n = 30
h = (bb-aa)/n
alfa = 0.00000001
number = 4   # Сколько графиков и альф хотите

x = [i*h for i in range(n+1)]
```

```
K = lambda x,t: 1/(3/2+x+t)**2
f = lambda t: log(3*(2*t + 3)/(2*t + 5))/(1 + t) - log(2)/(t + 3/2) - log(3/2)/(t + 5/2)

phi = lambda i,t: cos(i*acos(t))
right = lambda t,a: K(a,t)*f(t)
left = lambda t,y,i,x: K(x,y)*K(y,t)*phi(i,t)
func = lambda t,i,x: K(x,t)*phi(i,t)

def solution(alfa):
    CC = np.zeros((n+1, n+1))
    F = np.zeros(n+1)
    for j in range(n+1):
        F[j]= quad(right,0,1, args=(x[j]))[0]
        for i in range(n+1):
            CC[j][i]=alfa*phi(i,x[j]) + dblquad(left,0,1,0,1, args=(i,x[j]))[0]
    C = np.linalg.solve(CC, F)

    eps = np.zeros(n+1)
    S = np.zeros(n+1)
    for j in range(n+1):
        for i in range(n+1):
            S[j] = S[j] + C[i]*quad(func,0,1,args=(i,x[j]))[0]
        eps[j] = abs(f(x[j]) - S[j])
        # print('error for i =', j, 'steps =  ',eps)
    print('errors = ',eps)
    U = np.zeros(n+1)
    for j in range(n+1):
        for i in range(n+1):
            U[j] = U[j] + C[i]*phi(i,x[j])
    return U

def pltplot(l,x, alfa):   # Рисуут графики с шагом alfa*10
    al = alfa*(10**l)
    print('Alfa = ', al)
    A = solution(al)
    plt.plot(x, A, label = 'alfa = ' + str(al))
    print("_____________________________________")

def main():
    fig, ax = plt.subplots()
    for i in range(number):
        pltplot(i,x,alfa)

    ax.set(xlabel='x', ylabel='U')
    ax.legend()
    ax.grid()

    fig.savefig("test.png")
    plt.show()

main()
```

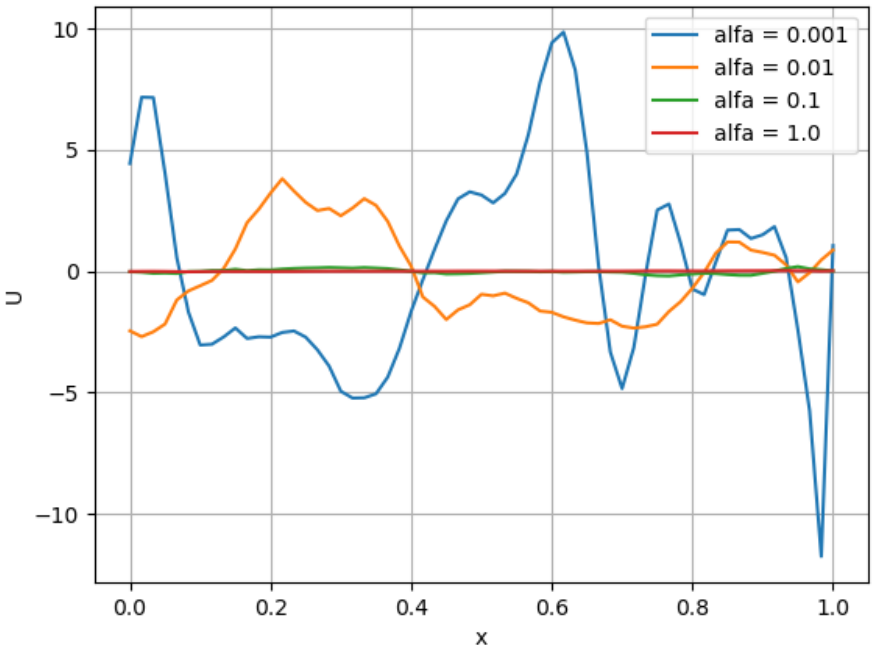
Результаты: Давайте проверим точность результатов. Выведем ошибку для каждого альфа для разных n:

```
n = 30:

Alfa =  0.001
errors =  [0.01114158 0.01113227 0.01110915 0.01107376 0.01102753 0.01097179
 0.01090769 0.01083619 0.01075838 0.01067488 0.01058661 0.01049414
 0.01039814 0.01029912 0.01019751 0.01009385 0.00998848 0.00988172
 0.0097739  0.00966531 0.00955622 0.0094468  0.0093373  0.00922788
 0.0091187  0.00900988 0.00890158 0.00879389 0.00868692 0.00858076
 0.00847544]
-------------------------------------
Alfa =  0.01
errors =  [0.03781959 0.03669689 0.03562589 0.03460301 0.03362528 0.03269009
 0.0317949  0.03093698 0.03011485 0.02932581 0.02856843 0.02784083
 0.02714152 0.02646891 0.02582145 0.02519818 0.02459778 0.02401902
 0.02346081 0.02292222 0.02240237 0.02190025 0.02141521 0.02094632
 0.02049298 0.02005434 0.01962988 0.01921898 0.01882102 0.01843547
 0.01806171]
-------------------------------------
Alfa =  0.1
errors =  [0.00384973 0.00442962 0.00494255 0.00539564 0.00579518 0.00614677
 0.00645537 0.00672536 0.00696073 0.00716492 0.00734111 0.00749209
 0.00762036 0.00772818 0.00781756 0.00789034 0.00794815 0.00799245
 0.00802459 0.00804575 0.00805703 0.00805941 0.00805378 0.00804092
 0.00802159 0.00799644 0.00796606 0.00793101 0.00789177 0.00784879
 0.00780249]
-------------------------------------
Alfa =  1.0
errors =  [0.03471389 0.03310595 0.03160163 0.03019244 0.02887075 0.02762967
 0.02646295 0.02536493 0.02433048 0.02335493 0.022434   0.02156382
 0.02074082 0.01996175 0.01922363 0.01852372 0.0178595  0.01722866
 0.01662906 0.01605872 0.01551584 0.01499872 0.0145058  0.01403565
 0.01358691 0.01315836 0.01274882 0.01235722 0.01198256 0.0116239
 0.01128038]
```




```
n = 60:

Alfa =  0.001
errors =  [0.00112406 0.00680337 0.00674728 0.00502965 0.0055835  0.004069
 0.00094525 0.00435066 0.00535393 0.00585075 0.00723353 0.00359822
 0.00502988 0.00440065 0.00595916 0.00398513 0.0072524  0.00396283
 0.00404401 0.00576612 0.00572104 0.00434448 0.006366   0.00521167
 0.00687904 0.00487894 0.00592901 0.00679241 0.00619856 0.00616813
 0.00551809 0.00668012 0.00319054 0.00246932 0.00447237 0.00381479
 0.00601174 0.00507951 0.00355975 0.00396635 0.00420839 0.00382406
 0.00448546 0.00450658 0.00487727 0.00443996 0.00392898 0.00576587
 0.00428865 0.00215384 0.00413276 0.00401525 0.00466451 0.00173459
 0.00423881 0.00336875 0.00355573 0.00439158 0.00386666 0.00439515
 0.0033681 ]
-----------------------------------
Alfa =  0.01
errors =  [1.99095866e-02 3.01940471e-02 2.49481219e-02 2.00089098e-02
 6.57309678e-03 1.23847129e-02 1.70624700e-03 2.94732678e-02
 1.37954815e-02 1.71634206e-02 1.61514542e-02 1.02342084e-02
 2.02520236e-03 6.34542798e-03 2.56304795e-02 1.65614523e-02
 6.33058919e-04 3.25508413e-03 1.52739944e-02 8.40618642e-03
 8.19393504e-03 9.48961608e-03 1.90766866e-02 1.20606256e-02
 5.81927002e-03 8.27018062e-03 3.52417782e-03 7.90193535e-04
 2.04036389e-02 1.16948285e-02 1.48275674e-02 8.11517212e-03
 5.06256982e-04 2.55197622e-03 2.90812820e-03 1.69510394e-03
 4.66051355e-04 6.40933653e-05 6.91176125e-04 3.00328018e-03
 2.35854215e-03 2.50044191e-04 4.59996247e-03 2.87402478e-03
 2.38587225e-03 1.10963886e-03 3.32597003e-03 2.31484532e-03
 7.34286975e-04 3.69399896e-03 1.70385388e-04 3.93543822e-03
```

```
 1.99617635e-03 9.30663067e-03 2.47366406e-03 6.80668733e-03
 3.03269210e-03 2.83234609e-03 2.39830440e-03 2.42257351e-03
 1.90316451e-03]
-------------------------------------
Alfa =  0.1
errors =  [0.03695022 0.03613435 0.03177067 0.03158688 0.03470608 0.03636203
 0.03498903 0.02960606 0.03243679 0.02614192 0.02620495 0.02994674
 0.02974371 0.02920086 0.02462923 0.02508283 0.0284714  0.02742351
 0.02320959 0.02484384 0.02396302 0.02551198 0.01936577 0.02178806
 0.02228741 0.02206949 0.02212808 0.02097986 0.0158942  0.01699434
 0.01959967 0.01691372 0.02040819 0.01922132 0.01890565 0.0199357
 0.01785673 0.01866421 0.01471477 0.01539628 0.01595829 0.01598165
 0.01454557 0.01442863 0.01674198 0.01386691 0.01363637 0.01456243
 0.01523969 0.01512009 0.01388329 0.01155582 0.01275727 0.01480077
 0.01246846 0.01503248 0.01308757 0.01236202 0.01325684 0.01284084
 0.01200922]
-------------------------------------
Alfa =  1.0
errors =  [0.03770854 0.03685066 0.03619657 0.03520928 0.03436404 0.03367899
 0.03281977 0.03238135 0.03149236 0.03092493 0.03029822 0.02946814
 0.02876192 0.02833749 0.02801884 0.02726105 0.0266103  0.026044
 0.02578273 0.02515553 0.02469365 0.02416807 0.02386839 0.02335372
 0.02285199 0.0224929  0.02198816 0.02156752 0.02135306 0.02095078
 0.02050254 0.02024163 0.01966576 0.01940217 0.01903956 0.01871012
 0.01836989 0.01807671 0.01791266 0.01754452 0.01720479 0.01688752
 0.01673255 0.01647491 0.01610601 0.01590798 0.01566309 0.0153596
 0.01513316 0.01493192 0.01472209 0.01453005 0.01426469 0.01395727
 0.01385355 0.01352786 0.01343788 0.01323224 0.01294995 0.01283449
 0.01265908]
```



Отсюда выдно оптимально значение альфы. Оно равно 0.01