УДК 519.688
МРНТИ 28.17.1

METHODS OF MODERNIZING THE APPEAL OF LAPLACE TRANSFORMATION IN THE PYTHON LANGUAGE TO SOLVE ELECTRICAL CIRCUITS

A.M. ABEUOVA

International University of Information Technologies

Abstract: *In applied mathematics, the Laplace transform is very relevant. Thus, in mathematics, mechanics and engineering, the operational method based on the Laplace transform is widely and very successfully used to solve entire class of problems. The object of the study is the integral from the Riemann-Mellin formula for inverting the Laplace transform and approximating it with the help of Fourier series of a numerical aggregate. In this paper we study the method of inversion of the Laplace transform by expanding the original function into a Fourier series with respect to the sines of odd multiple arcs. Also an analogous method based on the expansion of the function in the Fourier series of Legendre polynomials is developed, during which examples were considered that helped to carry out a comparative analysis of the convergence rates of canonical and developed methods. A numerical apparatus was developed and implemented in the programming language Python, which clearly demonstrates the predicted hypotheses.*

Keywords: *Laplace Transformation, Image, Original, Asymptotics, OF-symbols, Fourier Series, Orthogonal Polynomials*

ЭЛЕКТР ТІЗБЕКТЕРІН ШЕШУ ҮШІН PYTHON ТІЛІНДЕ ЛАПЛАС ТҮРЛЕНДІРУІН ЖАҢҒЫРТУ ӘДІСТЕРІ

Аңдатпа: *Қолданбалы математикада Лапластың түрленуі өте өзекті. Осылайша математикада, механикада және техникада Лапласты түрлендіруге негізделген жұмыс әдісі есептің барлық класын шешу үшін кең және өте табысты қолданылады. Зерттеу объектісі – Лапласты түрлендіру және сандық агрегаттың Фурье қатарының көмегімен аппроксимациялау үшін Риман-Меллин формуласынан интеграл. Бұл мақалада біз бастапқы функцияны Фурье қатарына тақ еселік доғалардың синусы бойынша жіктеу жолымен Лапласты түрлендіру әдісін зерттейміз. Сонымен қатар, Фурье қатарына функцияны Лежандрдың көпмүшелерінен ажыратуға негізделген ұқсас әдіс әзірленді, оның барысында каноникалық және әзірленген әдістердің жинақталу жылдамдығына салыстырмалы талдау жүргізуге мүмкіндік берген мысалдар қарастырылды. Сандық аппарат болжамды гипотезаларды көрнекі көрсететін Python бағдарламалау тілінде әзірленген және жүзеге асырылған.*

Түйінді сөздер: *Лаплас түрлендіру, сурет, түпнұсқа, асимптотика, О-символдар, Фурье қатарлары, ортогоналды көпмүшелер*

МЕТОДЫ МОДЕРНИЗАЦИИ ОБРАЩЕНИЯ ПРЕОБРАЗОВАНИЯ ЛАПЛАСА НА ЯЗЫКЕ PYTHON ДЛЯ РЕШЕНИЯ УРАВНЕНИЙ ЭЛЕКТРИЧЕСКИХ ЦЕПЕЙ

Аннотация: *В прикладной математике преобразование Лапласа очень актуально. Таким образом, в математике, механике и технике метод работы, основанный на преобразовании Лапласа, широко и очень успешно используется для решения всего класса задач. Объектом исследования является интеграл от формулы Римана-Меллина для обращения преобразования Лапласа и аппроксимации его с помощью рядов Фурье числового агрегата. В этой статье мы изучаем метод обращения преобразования Лапласа путем разложения исходной функции в ряд Фурье по синусам нечетных кратных дуг. Также разработан аналогичный метод, основанный на разложении функции в ряды Фурье от многочленов Лежандра, в* *ходе которого были рассмотрены примеры, позволившие провести сравнительный анализ скоростей сходимости канонических и разработанных методов. Числовой аппарат был разработан и реализован на языке программирования Python, который наглядно демонстрирует предсказанные гипотезы.*

Ключевые слова: *преобразование Лапласа, изображение, Оригинал, асимптотика, O-символы, ряды Фурье, ортогональные многочлены*

INTRODUCTION

In applied mathematics, the Laplace transform is very relevant. Thus, in mathematics, mechanics and engineering, the operational method based on the Laplace transform is widely and very successfully used to solve entire class of problems. In many cases of using this transformation the most difficult is its conversion.

Let us recall some basic definitions from the theory of the Laplace transform [1]. Suppose that a Riemann integrable function $f(t)$ is given on the $0 \leq t < \infty$ semi-axis. Then the Laplace transform of the function $f(t)$ is determined by the equality (1)

$$F(p) = \int_0^\infty f(t)e^{-pt}\,dt\,. \quad (1)$$

The function $f(t)$ is called the original, and the function $F(p)$ is the image. The fact that this is the Laplace transform of a function will be described as follows:

$F(p) = \mathcal{L}\{f(t)\}(p)$.

The operational method of solving problems is usually divided into 4 stages:

1. From the required original function $f(t)$, they go to the image function $F(p)$;

2. Above the image $F(p)$ operations are performed corresponding to the operations above $f(t)$, after which an equation is obtained for, which is often much simpler than the equation for the originals;

3. The resulting equation for the images is solved relatively $F(p)$;

4. From the found image $F(p)$ go to the original $f(t)$, which is the desired function.

The last step of this algorithm is the most difficult part because of the hardness of computing the integral (1). So, this paper will study this step more detailed with some suggestions of improvements.

The inversion method of the Laplace transform using the decomposition of the original in a Fourier series.

In the future, we will use the identity (2) from [2]:

$$\cos^{2n}\theta\sin\theta = \frac{1}{2^{2n}}\sum_{k=0}^{n}\left(C_{2n}^{k} - C_{2n}^{k-1}\right)\sin\big((2(n-k)+1)\theta\big). \quad (2)$$

The considered method is built on two assumptions that increase the rate of convergence, but not limiting its generality:

• $f(0) = 0$;

• The image $F(p)$ exists, if $Re\,p > 0$.

The Laplace integral (1) is transformed by replacing (3)

$$e^{-t} = \cos\theta. \quad (3)$$

We introduce the notation (4)

$$f(t) = f(-\ln\cos\theta) = \varphi(\theta),$$
$$0 \leq \theta < \frac{\pi}{2} \quad (4)$$

Applying the given notation, we arrive at the following identity (5):

$$F(p) = \int_0^{\frac{\pi}{2}} \varphi(\theta)\cos^{p-1}\theta\sin\theta\,d\theta. \quad (5)$$

So, now our goal is to find the function $\varphi(\theta)$, that can easily help us to restore the original function $f(t)$.

We can decompose the function $\varphi(\theta)$ into a Fourier series in sines of odd multiple arcs, as in (6):

$$\varphi(\theta) = \sum_{k=0}^{\infty} c_k \sin\big((2k+1)\theta\big). \quad (6)$$

Coefficients c_k are determined by usual way (7):

$$c_k = \frac{4}{\pi}\int_0^{\frac{\pi}{2}} \varphi(\theta)\sin\big((2k+1)\theta\big)\,d\theta. \quad (7)$$

In order to express the value of the coefficients c_k through the image function's values $F(p)$, we will use one more replacement $p = 2n+1$ $(n = 0, 1, 2, \ldots)$. Then we will get (8):

$$F(2n+1) = \int_0^{\frac{\pi}{2}} \sum_{k=0}^{\infty} c_k \sin\big((2k+1)\theta\big)\cos^{2n}\theta \sin\theta\, d\theta. \quad (8)$$

In this step, we use identity (9). Then consider the integral, which we obtain in the course of transformations (9):

$$\int_0^{\frac{\pi}{2}} \sin\big((2\mu+1)\theta\big)\sin\big((2\nu+1)\theta\big)\,d\theta = \begin{cases} 0, & \mu \neq \nu \\ \dfrac{\pi}{4}, & \mu = \nu \end{cases}. \quad (9)$$

Based on this, with a fixed *n* there are only those values, that are $\nu = n - k\,(k = 0,1,2\ldots n)$. As the result, the Laplace formula can be presented as in (10):

$$F(2n+1) = 2^{-2n}\frac{\pi}{4}\sum_{k=0}^{n}\left(C_{2n}^{k} - C_{2n}^{k-1}\right)c_{n-k}. \quad (10)$$

Substituting this equality sequentially $n = 0,1,2\ldots$, we can get the linear system of get a linear system of equations (11) with a triangular matrix to determine the coefficients c_k:

$$\begin{bmatrix} 1 & 0 & \cdots & 0 \\ 1 & 1 & & 0 \\ \vdots & & \ddots & \vdots \\ \dfrac{1}{2n+1}C_{2n+1}^{n} & \dfrac{3}{2n+1}C_{2n+1}^{n-1} & \cdots & 1 \end{bmatrix}\begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{pmatrix} = \begin{pmatrix} \dfrac{4}{\pi}F(1) \\ \dfrac{4^2}{\pi}F(3) \\ \vdots \\ \dfrac{4^{n+1}}{\pi}F(2n+1) \end{pmatrix}. \quad (11)$$

Next, we find the coefficients c_k, and the function $\varphi(\theta)$ is restored as the limit of the partial sums of the Fourier series (6).

The inversion method of the Laplace transform using the expansion of the original function in a Fourier series in Legendre polynomials

The author of the method studied in [3] does not explain several points in the inversion of the Laplace transform. Replacement $e^{-t} = \cos\theta$ in the Laplace integral is one of those moments. Let us try to apply a different replacement, easing perhaps the solution of the system and speeding up the rate of convergence of the method.

In the original Laplace integral (1) we make the replacement $e^{-t} = \theta$ instead of canonical cos. We introduce the notation $f(t) = f\left(-\frac{1}{\delta}\ln\theta\right) = \varphi(\theta)$. Then the image will be an integral (12)

$$F(p) = \int_0^1 \theta^{p-1}\varphi(\theta)d\theta. \quad (12)$$

As a result, we obtain that the task is to find a function $\varphi(\theta)$ through which it will be easy to express the original function.

Let us say, $p = n$, where $n = 1, 2, 3, \ldots$ then we get (13):

$$F(n) = \int_0^1 \theta^{n-1}\varphi(\theta)d\theta. \quad (13)$$

Recall that the Legendre polynomials [4] (14)

$$L_n(\theta) = (\theta^n(1-\theta)^n)^{(n)} \quad (14)$$

form a complete orthogonal system on [0,1], i.e. the scalar product of two different polynomials is zero (15):

$$\langle L_i, L_j\rangle = 0. \quad (15)$$

In this case, the Fourier expansion coefficients for the Legendre polynomials are calculated by the formula (16):

$$c_k = \frac{\langle L_k,\varphi\rangle}{\langle L_k,L_k\rangle}. \quad (16)$$

We decompose the two integrand factors by the Legendre polynomials (17):

$$\varphi(\theta) = \sum_{k=0}^{\infty} c_k L_k(\theta); \qquad \theta^{n-1} = \sum_{j=0}^{n-1} a_j L_j(\theta), \quad (17)$$

(a) (б)

where $a_j = \frac{\langle L_j,\theta^{n-1}\rangle}{\langle L_j,L_j\rangle}$. Next, we describe several simple transformations over the integral. Then substituting the above decompositions into the integral, we obtain the following equation (18):

$$F(n) = \int_0^1 \sum_{j=0}^{n-1} \frac{\langle L_j,\theta^{n-1}\rangle}{\langle L_j,L_j\rangle} L_j(\theta) \sum_{k=0}^{\infty} c_k L_k(\theta)\, d\theta \quad (18)$$

We take out the multiplier factors that are independent of the variable θ (19):

$$F(n) = \sum_{j=0}^{n-1}\sum_{k=0}^{\infty} \frac{\langle L_j,\theta^{n-1}\rangle}{\langle L_j,L_j\rangle} c_k\langle L_j,L_k\rangle. \quad (19)$$

Taking advantage of the orthogonality of the Legendre polynomials, we obtain that only the terms remain for $j = k$ (20).

$$F(n) = \sum_{j=0}^{n-1} \frac{\langle L_j,\theta^{n-1}\rangle}{\langle L_j,L_j\rangle} c_j\langle L_j,L_j\rangle = \sum_{j=0}^{n-1} c_j\langle L_j,\theta^{n-1}\rangle. \quad (20)$$

So, from this equality it is clear that the system for finding the coefficients of the expansion in a Fourier series in Legendre polynomials are the scalar products of the Legendre polynomials and the decomposed function. Find the dot product $\langle L_j, \theta^{n-1}\rangle$ in (21):

$$\langle L_j, \theta^{n-1}\rangle = \int_0^1 ((\theta-\theta^2)^j)^{(j)} \cdot \theta^{n-1} d\theta. \tag{21}$$

Having integrated this equality j times in pars, we rewrite it in the following form (22):

$$\langle L_j, \theta^{n-1}\rangle = (-1)^j \int_0^1 (\theta-\theta^2)^j \cdot (\theta^{n-1})^{(j)} d\theta. \tag{22}$$

It is easy to calculate the derivative of the jth order of a power function. Then, having performed simple transformations, we obtain the scalar product (23):

$$\langle L_j, \theta^{n-1}\rangle = (-1)^j \frac{(n-1)!}{(n-1-j)!} \int_0^1 (1-\theta)^j \cdot \theta^{n-1} d\theta. \tag{23}$$

Note that the integral is a beta function $B(x,y) = \frac{\Gamma(x)\Gamma(y)}{\Gamma(x+y)}$, where $\Gamma(x)$ is a gamma function. Thus, we can rewrite the answer in (24):

$$\langle L_j, \theta^{n-1}\rangle = (-1)^j \frac{(n-1)!}{(n-1-j)!} B(n, j+1). \tag{24}$$

After we write the beta function through factorials, it remains to substitute the resulting expression into the Laplace integral. Thus, the system (25) is obtained:

$$F(n) = \sum_{j=0}^{n-1} c_j (-1)^j \frac{(n-1)!^2}{(n-1-j)!} \cdot \frac{j!}{(n+j)!}, \tag{25}$$

where $n = 1,2,3,\ldots$

You can write this system in matrix form (26):

$$\begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0.5 & -1 & & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{n} & (-1)\frac{(n-1)!^2}{(n-2)!} \cdot \frac{1}{(n+1)!} & \cdots & (-1)^{n-1}\frac{(n-1)!^3}{(2n-1)!} \end{bmatrix} \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{pmatrix} = \tag{26}$$
$$= \begin{pmatrix} F(1) \\ F(3) \\ \vdots \\ F(2n+1) \end{pmatrix}.$$

Note that the matrix in (27)

$$A = \begin{bmatrix} 1 & & 0 & \cdots & & 0 \\ 0.5 & & -1 & & & 0 \\ \vdots & & \vdots & \ddots & & \vdots \\ \frac{1}{n} & (-1)\frac{(n-1)!^2}{(n-2)!} \cdot & \frac{1}{(n+1)!} & \cdots & (-1)^{n-1} & \frac{(n-1)!^3}{(2n-1)!} \end{bmatrix} \quad (27)$$

has an inverse lower triangular matrix, which allows us to solve this system without additional difficulties.

Having solved this system, we find the Fourier series coefficients by the Legendre polynomials for the function $\varphi(\theta)$, by which it is easy to reconstruct the image function $f(t)$.

An empirical view on the inversion method of the Laplace transform using the expansion of a function in a Fourier series in Legendre polynomials

Let us give an example of the application of this method in practice for the image function $F(p) = \frac{1}{p^2}$. This image corresponds to the original $f(t) = t$.

Let us compare the result of the method of inversion of the Laplace transform with the original function $f(t) = t$. Although the expansion coefficients in a series, calculated using a numerical solution of the system almost completely coincide with the actual decomposition coefficients, and besides, in the method using Legendre orthogonal polynomials, the formulas themselves visually looked simpler than in the method of conversion using the original the sines of odd multiple arcs, in practice, there was a problem with the fact that the computer does not build graphs of high order badly. And therefore, even the almost exact finding of the highest coefficients does not help to accurately depict the graph of the partial amount.

So, on fig. 1 shows that for $n = 50$ and $n = 10$ the recovery accuracy of the original is noticeably different.

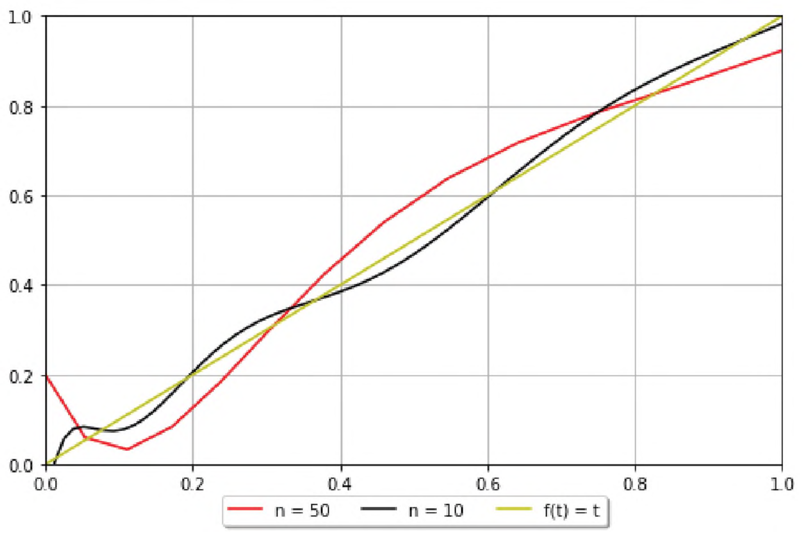

Figure 1 – The result of Laplace transform inversion in Legendre polynomials for original f(t)=t

Namely, for a small order n, the recovery was even more accurate. Therefore, although theoretically this method is not bad, it is practically very difficult to implement.

Investigate empirically the convergence rate parameter of the method (2.28)

$$\alpha = -\frac{\ln(f(t) - \sum_{k=0}^{n} c_k L_k(e^{-t}))}{\ln n}. \quad (28)$$

The graph of the behavior of this parameter is presented in Fig.2.

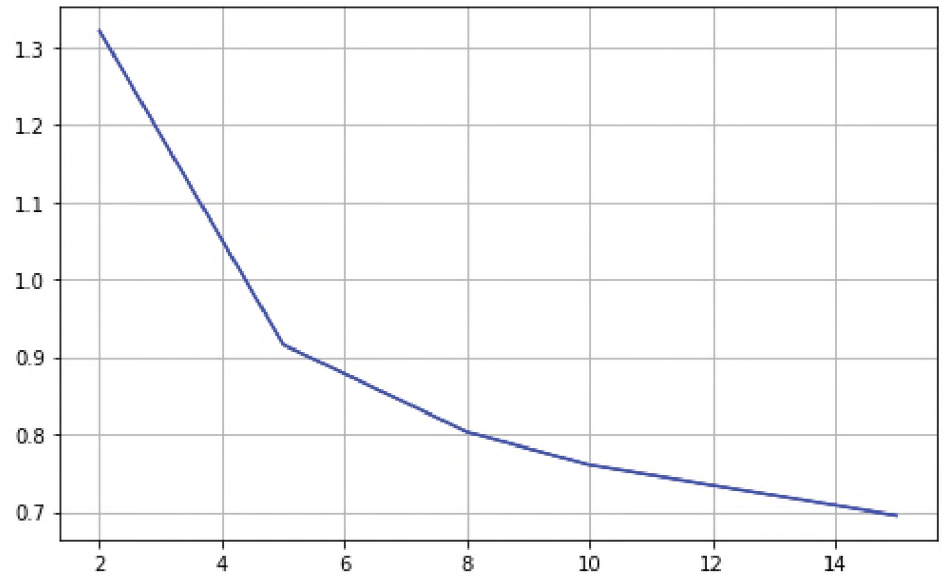

Figure 2 – Behaviour of α

As you can see, with an increase in the number n, the parameter α tends to zero, which confirms our theoretical estimates — the method diverges for large n.

CONCLUSION

Thus, a numerical apparatus was built for calculating the inversion of the Laplace transform in a well-known image using the expansion of the function in a Fourier series in Legendre polynomials, which has been tested in practice. It can be concluded that the use of polynomials in practice does not always work (especially high-order polynomials).

REFERENCES

1. Glushko A. (2004), Laplace tranformation. Properties and applications. Voronezh, pp. 59
2. Philatov A., Kolpakov A., Philatov A. (2006), Inverse Laplace transform, LAP LAMBERT Academic Publishing, pp. 64
3. Немыцкий, В.В., 1948. ГМ Фихтенгольц,“Курс дифференциального и интегрального исчисления, т. I”(рецензия). Успехи математических наук, *3*(4 (26), pp.181-183.
4. Владимиров, В.С., 2004. Уравнения математической физики: Учеб. для студентов вузов. ВС Владимиров, ВВ Жаринов-М.: ФИЗМАТЛИТ.

APPLICATION. CODE ON PYTHON FOR LAPLACE TRANSFORM INVERSION

```
import matplotlib.pyplot as plt
import numpy as np
import math
from scipy.misc import derivative
from scipy.special import factorial

from scipy.special import binom
import scipy.integrate as integrate
from scipy.special import gamma
from sympy import (
            symbols, sqrt, exp, sin,
            diff, pi
        )
def replace(tetta):
  return (-1)*math.log(math.cos(tetta))

def F(n):
  return gamma(n+1)*gamma(1/2)/(gamma(n+1/2+1)*2*(2*n+1))
def FD(n):
  return gamma((2*n+1)*delta/2+1/2)*gamma(1/2)/(gamma((2*n+1)*delta/2+1)*2*(2*n+1)*d
elta)

def matrixA(n):
  A = np.zeros((n, n))
  for k in range(n):
    for m in range(n):
      A[k][m] = (2*m+1)*binom(2*k+1, k-m)/(2*k+1)
  return A

def columnB(n):
  B = np.zeros(n)
  for i in range(0, n):
    B[i] = 4**(i+1)*F(i)/np.pi
  return B

n = 23
nn=100
f = []
fOrigin = []
tx = []
tetta = np.linspace(0, np.pi/2, nn)



coefficients = np.linalg.solve(matrixA(n), columnB(n))
s = 0
```

```
for i in range(nn):
   s = 0
   tx.append(replace(tetta[i]))
   for j in range(0, n):
      s += coefficients[j]*math.sin((2*j+1)*tetta[i])
   f.append(s)
   fOrigin.append(math.acos(math.exp((-1)*tx[i])))




fig, ax = plt.subplots(figsize=(8, 5))
ax.plot(tetta, tetta, color = 'b', label = 'Original function')
ax.plot(tetta, f, color = 'y', label = 'A system solution')

legend = ax.legend(loc='upper center', bbox_to_anchor=(0.5, -0.05),
          fancybox=True, shadow=True, ncol=5)
plt.xlim(0, np.pi/2)
plt.grid()
plt.show()

fig, ax = plt.subplots(figsize=(8, 5))
ax.plot(tx, f, color = 'r', label ='f(t) = arccos(exp(-t))')
ax.plot(tx, fOrigin, color = 'g', label ='f(t) = arccos(exp(-t))')

legend = ax.legend(loc='upper center', bbox_to_anchor=(0.5, -0.05),
          fancybox=True, shadow=True, ncol=5)
#plt.ylim(0, 2)
plt.xlim (0, np.pi/2)
plt.grid()
plt.show()
```