Journal of Applied Analysis and Computation
Volume 14, Number 1, February 2024, 317–333
Website:http://www.jaac-online.com
DOI:10.11948/20230222

NEW EFFECTIVE TRANSFORMATIONAL COMPUTATIONAL METHODS

Jun Zhang[1,†], Ruzong Fan[2], Fangyang Shen[3] and Junyi Tu[4]

Abstract Mathematics serves as a fundamental intelligent theoretic basis for computation, and mathematical analysis is very useful to develop computational methods to solve various problems in science and engineering. Integral transforms such as Laplace Transform have been playing an important role in computational methods. In this paper, we will introduce Sumudu Transform in a new computational approach, in which effective computational methods will be developed and implemented. Such computational methods are straightforward to understand, but powerful to incorporate into computational science to solve different problems automatically. We will provide computational analysis and essentiality by surveying and summarizing some related recent works, with additional automatic proof details by applying system built-in functions. Applications include the computation of coefficients of Taylor's expansions, calculation of generating functions, mathematical identity proofs, solving differential equations and integral equations. For demonstration purposes, some of the methods were implemented in Maple with demonstrational results matching the expected values.

Keywords Sumudu transform, automatic calculation, computation of coefficient, generating function, identity proof, differential equation, integral equation, problem solving.

MSC(2010) 33F10, 34-04, 40-04, 44-04, 44A10, 44A30, 45-04.

1. Introduction of Sumudu Transform

Mathematical analysis is very useful to develop computational methods to solve various problems in the real world. Named after its inventor Pierre-Simon Laplace, Laplace Transform is an integral transform with many applications in science and engineering. Since it is very famous and widely used, many undergraduate and graduate programs offer courses on the Laplace Transform.

The Sumudu Transform is recent, but it is as powerful as the Laplace Transform, furthermore, it has many nice new features. Laplace Transform is well studied and

†The corresponding author.
[1]Department of Computer Science and Engineering Technology, University of Maryland Eastern Shore, Princess Anne, MD 21853, USA
[2]Department of Biostatistics, Bioinformatics, and Biomathematics, Georgetown University, Washington, DC 20057, USA
[3]Department of Computer Systems Technology, New York City College of Technology, CUNY, Brooklyn, NY 11201, USA
[4]Department of Computer Science, Salisbury University, Salisbury, MD 21801, USA
Email: jzhang@umes.edu(J. Zhang), rf740@georgetown.edu(R. Fan), fshen@citytech.cuny.edu(F. Shen), jxtu@salisbury.edu(J. Tu)

implemented in computer algebra systems such as Maple. Since the similarity of these two transforms, it is a good idea to introduce the Sumudu Transform in comparison with Laplace Transform.

Let f be a function of variable x defined in the domain of $0 <= x < \infty$. The Laplace Transform and Sumudu Transform of f can be defined respectively as:

$$G(s) = L[f(x)] = \int_0^\infty e^{-xs} f(x)dx,$$
$$H(s) = S[f(x)] = \int_0^\infty \frac{1}{s} e^{-x/s} f(x)dx.$$

We refer to $f(x)$ as the original function of $G(s)$ or $H(s)$ respectively, $G(s)$ as the Laplace Transform of the function $f(x)$, $H(s)$ as the Sumudu Transform of the function $f(x)$. We also refer to $f(x)$ as the inverse Laplace Transform of $G(s)$ or inverse Sumudu Transform of $H(s)$. The symbol L denotes the Laplace Transform, the exponential function e^{-xs} is called the kernel of the Laplace Transform. The symbol S denotes the Sumudu Transform, the function $\frac{1}{s}e^{-x/s}$ is called the kernel of the Sumudu Transform.

Sumudu Transform was introduced and studied in a traditional way as other integral transforms by many researchers recently [1–5, 7–9, 11, 15, 23], it was studied through new computational approaches [17–22]. In this paper, we will introduce the essentiality and analysis of the transform in computational approaches, which can be used to solve various problems automatically.

Since both Laplace and Sumudu Transforms are defined as improper integral, we have the basic properties based on the integral calculation as following:

Linearity:

$$L[c_1 f(x) + c_2 g(x)] = c_1 L[f(x)] + c_2 L[g(x)],$$
$$S[c_1 f(x) + c_2 g(x)] = c_1 S[f(x)] + c_2 S[g(x)].$$

Shift:

$$L[e^{ax} f(x)] = G(s - a),$$
$$S[e^{ax} f(x)] = \frac{1}{1 - as} H(\frac{s}{1 - as}).$$

Scaling:

$$L[f(ax)] = G(s/a)/a,$$
$$S[f(ax)] = H(as).$$

Derivative:

$$L[f^m(x)] = s^m L[f(x)] - s^{m-1} f(0) - \cdots - f^{m-1}(0),$$
$$S[f^m(x)] = S[f(x)]/s^m - f(0)/s^m - \cdots - f^{m-1}(0)/s.$$

Integral:

$$L[\int_0^x f(t)dt] = L[f(x)]/s,$$

$$S[\int_0^x f(t)dt] = sS[f(x)].$$

Convolution:

$$L[(f * g)(x)] = L[f(x)]L[g(x)],$$
$$S[(f * g)(x)] = sS[f(x)]S[g(x)].$$

Laplace-Sumudu Duality:

$$H(s) = G(1/s)/s,$$
$$G(s) = H(1/s)/s.$$

Notice the similarities of the properties for both transforms. These properties are very useful for problem solving, including automatic coefficient calculation, automatic calculation of generating functions, automatic proof of identities, automatic solving of differential and integral equations, etc.

A very surprising and useful fact is that the Sumudu Transform and its original function share the same Taylor's coefficients except for a factor of $n!$. This fact can be explained by the following theorems:

Theorem 1.1 ([5]). *If function $f(x)$ can be expanded as:*

$$f(x) = \sum_{n=0}^{\infty} a_n x^n,$$

then, its Sumudu Transform can be written as the power series function:

$$S[f](x) = \sum_{n=0}^{\infty} n! a_n x^n.$$

Theorem 1.2 ([16]). *We assume that $f(x)$ is a bounded and continuous function of x and its Sumudu Transform satisfies:*

$$S[f](x) = \sum_{n=0}^{\infty} a_n x^n,$$

then we can expend:

$$f(x) = \sum_{n=0}^{\infty} a_n \frac{x^n}{n!}.$$

Theorem 1.1 and Theorem 1.2 give a complete relationship about coefficients under transforms. This is an important relationship. Based on this relationship, a lot of work can be done, including but not limited to the works in the following sections.

2. Coefficient calculation

A very surprising and useful fact regarding Sumudu Transform is that the original function and its Sumudu Transform share the same coefficients with an exception of a factor of $n!$, therefore we can calculate the nth coefficient a_n of a function f as following:

Algorithm 2.1

(I) Input function: $f(x)$.
(II) Compute the Sumudu Transform $S[f](x)$.
(III) Compute the nth coefficient $a_n = [x^n]S[f]$.
(IV) Return: $a_n/n!$.

We implement a simplified version of this algorithm in the Maple systems as the following *coeffCalculation* procedure.

```
with(inttrans);
with(genfunc);
coeffCalculation := proc(f, x)
  local tem;
  tem := laplace(f, x, y);
  tem := subs(y = 1/z, tem)/z;
  tem := rgf_expand(tem, z, n));
  tem := simplify((tem)/n!);
  return tem
end   proc
```

Unfortunately, in Maple, there is no such a procedure to calculate the general term of coefficient, but it has the *series* to calculate the first few terms. We can easily make comparisons as the following:

$> coeffCalculation(exp(x), x);$

$$\frac{1}{n!}$$

$> series(exp(x), x = 0, 8);$

$$1 + x + \frac{1}{2}x^2 + \frac{1}{6}x^3 + \frac{1}{24}x^4 + \frac{1}{120}x^5 + \frac{1}{720}x^6 + \frac{1}{5040}x^7 + O(x^8)$$

$> coeffCalculation(\sin(x), x);$

$$\frac{\sin\frac{n\pi}{2}}{n!}$$

$> series(\sin(x), x = 0, 8);$

$$x - \frac{1}{6}x^3 + \frac{1}{120}x^5 - \frac{1}{5040}x^7 + O(x^9)$$

$> coeffCalculation(\cos(x), x);$

$$\frac{\cos\frac{n\pi}{2}}{n!}$$

$> series(\cos(x), x = 0, 8);$

$$1 - \frac{1}{2}x^2 + \frac{1}{24}x^4 - \frac{1}{720}x^6 + O(x^8)$$

$> coeffCalculation(\cos(10x)^4, x);$

$$\frac{(-40I)^n + (40I)^n + 4(-20I)^n + 4(20I)^n}{16n!}$$

$> series(\cos(10x)^4, x = 0, 8);$

$$1 - 200x^2 + \frac{50000}{3}x^4 - \frac{6800000}{9}x^6 + O(x^8)$$

Algorithm 2.2

(I) Input function: $f(x)$.
(II) Compute $G(x)$, the inverse Sumudu Transform of $f(x)$.
(III) Compute the nth coefficient $a_n = [x^n]G(x)$.
(IV) Return: $a_n * n!$.

The calculation of inverse Sumudu Transform is not easy, so this algorithm is more theoretical, but less practical.

3. Generating functions

Generating functions are a very powerful tool in discrete mathematics and algorithm analysis. For a sequence $\{a_n, n = 0, ..., \infty\}$, there are two types of widely used generating functions:

$$g(x) = \sum_{n=0}^{\infty} a_n x^n,$$
$$h(x) = \sum_{n=0}^{\infty} \frac{a_n}{n!} x^n.$$

We call $g(x)$ and $h(x)$ the ordinary generating function and the exponential generating function of the sequence a_n respectively. According to Theorem 1.1 and Theorem 1.2 stated above, we can claim: the Sumudu Transform of an exponential generating function is its ordinary generating function; the inverse Sumudu Transform of an ordinary generating function is its exponential generating function. The relationship of these two widely used generating functions is nothing but exactly the Sumudu Transform.

These theorems serve as a base to calculate the general terms of Taylor's series expansions and to calculate the generating functions.

For a sequence $\{a_n, n = 0, ..., \infty\}$, the following algorithm will print out the exponential generating function and return the ordinary generating function:

Algorithm 3.1

(I) Input sequence $\{a_n, n = 0, ..., \infty\}$.
(II) Compute and print out $h(x) = \sum_{n=0}^{\infty} \frac{a_n}{n!} x^n$, the the exponential generating function.
(III) Compute the Laplace Transform of $h(x)$: $G(x) = L[h(x)]$.
(IV) Return the ordinary generating function $g(x) = G(1/x)/x$.

We are not attempting to implement Algorithm 3.1 completely and perfectly. Since Laplace Transform is implemented in algebra systems, a simplified demonstration version is implemented as following:

$with(inttrans)$;
$with(genfunc)$;
$genFunction := proc(coeff, x)$
$simplify(coeff * x^n/n!)$;
$sum(\%, n = 0..\infty)$;
$simplify(\%)$;
$print(\%)$;
$laplace(\%, x, s)$;
$subs(s = 1/t, \%)/t$;
$subs(t = x, \%)$;
$simplify((\%))$;
$return\ \%$
$end\quad proc$

We can make some tests.

For sequence $(a_n = 1, n = 0...\infty)$, to calculate the exponential generating function and ordinary generating function:

$> genFunction(1, x)$;
$> e^x$;
$> \frac{1}{1-x}$;

For sequence $(a_n = (-1)^n, n = 0...\infty)$, to calculate the exponential generating function and ordinary generating function:

$> genFunction((-1)^n, x)$;
$> e^{-x}$;
$> \frac{1}{1+x}$;

For sequence $(a_n = n, n = 0...\infty)$, to calculate the exponential generating function and ordinary generating function:

$> genFunction(n, x)$;
$> xe^x$;
$> \frac{x}{(1-x)^2}$;

For sequence $(a_n = c^n, n = 0...\infty)$, to calculate the exponential generating function and ordinary generating function:

$> genFunction(c^n, x)$;
$> e^{cx}$;
$> \frac{1}{1-cx}$;

For sequence $(a_n = \sin(\frac{n\pi}{2}), n = 0...\infty)$, to calculate the exponential generating function and ordinary generating function:

$> genFunction(\sin(\frac{n\pi}{2}), x)$;
$> \sin(x)$;
$> \frac{x}{1+x^2}$;

For sequence $(a_n = \cos(\frac{n\pi}{2}), n = 0...\infty)$, to calculate the exponential generating function and ordinary generating function:

$> genFunction(\cos(\frac{n\pi}{2}), x)$;
$> \cos(x)$;
$> \frac{1}{1+x^2}$;

For sequence $(a_n = 1+(-1)^n, n = 0...\infty)$, to calculate the exponential generating function and ordinary generating function:

$> genFunction(1 + (-1)^n, x)$;
$> e^x + e^{-x}$;
$> \frac{2}{1-x^2}$.

4. Proof of identities

We all know that $\sin^2(x)+\cos^2(x) = 1$ and this is the famous Pythagorean Identity. How to prove it automatically? Let $f(x) = \sin^2(x) + \cos^2(x) - 1$, we can simply apply *coeffCalculation* presented above easily to calculate a_n, the nth coefficient of $f(x)$. It happens that $a_n = 0$ for all $n = 0, 1, 2, 3, \ldots$. Since $f(x)$ is analytic, so $f(x) = 0$ for all x. This proves the identity automatically.

New let $f(x) = \sin(x) - (e^{Ix} - e^{-Ix})/2I$, we can simply apply *coeffCalculation* presented above to calculate a_n, what happens is that $a_n = 0$ for all non-negative integers n. Since $f(x)$ is analytic, so $f(x) = 0$ for all x. This proves the identity $\sin(x) = (e^{Ix} - e^{-Ix})/2I$ automatically. A lot of identities can be proved in the same way, and here are some samples:

$$\cos(x) = (e^{Ix} + e^{-Ix})/2,$$

$$\sin(x) = (e^{Ix} - e^{-Ix})/2I,$$

$$\cosh(x) = (e^{x} + e^{-x})/2,$$

$$\sinh(x) = (e^{x} - e^{-x})/2,$$

$$\cosh^2(x) - \sinh^2(x) = 1,$$

$$e^{ix} = \cos(x) + i\sin(x),$$

$$\sin^2(x) + \cos^2(x) = 1,$$

$$\cos(a + x) = \cos(a)\cos(x) - \sin(a)\sin(x),$$

$$\cos(a - x) = \cos(a)\cos(x) + \sin(a)\sin(x),$$

$$\sin(a + x) = \sin(a)\cos(x) + \cos(a)sin(x),$$

$$\sin(a - x) = \sin(a)\cos(x) - \cos(a)sin(x),$$

$$\cos(2x) = \cos^2(x) - \sin^2(x),$$

$$\cos(2x) = 2\cos^2(x) - 1,$$

$$\cos(2x) = 1 - 2\sin^2(x),$$

$$\sin(2x) = 2\cos(x)\sin(x),$$

$$\cos(3x) = \cos^3(x) - 3\sin^2(x)\cos(x),$$

$$\cos(3x) = 4\cos^3(x) - 3\cos(x),$$

$$\sin(3x) = -\sin^3(x) + 3\cos^2(x)\sin(x),$$

$$\sin(3x) = -4\sin^3(x) + 3\sin(x),$$

$$\cos(nx) = 2\cos(x)\cos((n-1)x) - \cos((n-2)x),$$

$$\sin(nx) = 2\cos(x)\sin((n-1)x) - \sin((n-2)x),$$

$$\cos(nx) + I\sin(nx) = (\cos(x) + I\sin(x))^n,$$

$$\cosh(nx) + \sinh(nx) = (\cosh(x) + \sinh(x))^n,$$

$$\sin^2(x) = (1 - \cos(2x))/2,$$

$$\sin^3(x) = (3\sin(x) - \sin(3x))/4.$$

5. Solving differential equations

First, let's review how to solve differential equations by Laplace Transform, then we will show the advantage of Sumudu Transform to solve such equations automatically without the calculation of inverse transform.

Now let's consider the differential equation with initial values:

$$\begin{aligned} &y'' + 2y' + y = 0,\\ &y(0) = 1,\\ &y'(0) = 2. \end{aligned}$$

Applying the Laplace Transform to both sides, we have

$$\begin{aligned} &L[y'' + 2y' + y] = L[0],\\ \Rightarrow &L[y''] + 2L[y'] + L[y] = 0,\\ &s^2L[y] - sy(0) - y'(0) + 2(sL[y] - y(0)) + L[y] = 0. \end{aligned}$$

Let $Y(s)$ denote $L[y(x)]$. Substituting with initial values, then

$$\begin{aligned} &s^2Y(s) - sy(0) - y'(0) + 2(sY(s) - y(0)) + Y(s) = 0,\\ &(s^2 + 2s + 1)Y(s) = s + 4,\\ \Rightarrow &Y(s) = \frac{s+4}{s^2+2s+1}. \end{aligned}$$

In the conventional way, we have to calculate the inverse transform. In Maple, by applying *invlaplace* with $Y(s)$, we have

$$y(x) = (1 + 3x)e^x.$$

To avoid the calculation of the inverse transform, we can use Laplace-Sumudu Duality to calculate the Sumudu Transform, we have:

$$F(s) = \frac{1+4s}{(1+s)^2}.$$

Since $F(s)$ is a rational function, we can just apply rgf_expand to expand $F(s)$ as:

$$F(s) = \sum_{n=0}^{\infty}(1-3n)(-1)^n s^n.$$

We can use Sumudu Transform property regarding coefficient calculation stated above, then

$$y(x) = \sum_{n=0}^{\infty} \frac{(1-3n)(-1)^n}{n!} x^n.$$

The correctness can be easily verified. One way is just applying the *coeffCalculation* procedure implemented above to $y(x) = (1+3x)e^{-x}$, we have

$$\frac{(1-3n)(-1)^n}{n!}.$$

Which shows automatically that the two solutions are the same since all corresponding coefficients are the same.

For rational functions, the calculation of the inverse Laplace Transforms can be done by using algebraic techniques, except that, in general, the computation of inverse Laplace Transforms is expensive and troublesome [6]. Since Sumudu Transform has similar properties of Laplace Transform, we can use Sumudu Transform to replace Laplace Transform, we have

$$\begin{aligned} & S[y'' + 2y' + y] = S[0], \\ \Rightarrow & S[y''] + 2S[y'] + S[y] = 0, \\ & \frac{S[y]}{s^2} - \frac{y(0)}{s^2} - \frac{y'(0)}{s} + 2\left(\frac{S[y]}{s} - \frac{y(0)}{s}\right) + S[y] = 0. \end{aligned}$$

Multiply s^2 to both sides and substitute with the initial values, we have

$$\begin{aligned} (s^2 + 2s + 1)S[s] &= 1 + 4s, \\ \Rightarrow S[y] &= \frac{1+4s}{s^2+2s+1}. \end{aligned}$$

We can solve the equation by calculating the inverse of Sumudu Transform, but we have a better way here, since $S[y]$ is a rational function, we can use *rgf_expand* to help us, then

$$S[y] = \sum_{n=0}^{\infty} (1-3n)(-1)^n x^n.$$

By taking the advantage of the Sumudu Transform property regarding coefficient calculation, we have

$$y(x) = \sum_{n=0}^{\infty} \frac{(1-3n)(-1)^n}{n!} x^n.$$

We can use either Laplace or Sumudu Transform to solve equations, but we recommend a good combination in the following method:

- Conducting Laplace Transform to transfer a differential equation to an algebraic equation.
- Solving the algebraic equation from the step above.
- Applying Laplace-Sumudu Duality to calculate the Sumudu Transform.
- Calculating the general term of coefficient a_n of the Sumudu Transform.

- Returning the solution which is the power series with $a_n/n!$ as the general term of coefficient.
- Optional, adding the power series summation to a close form.

The advantage of this method is that it avoids the inverse Laplace Transform calculation, especially when such an inverse calculation is too expensive or not possible.

In Maple, one can verify the correctness by comparing the solutions with Maple's $dsolve(deqns, vars, method = Transform)$ and set the method as $laplace$ with $dsolve(deqns, vars, method = Transform, options)$.

6. Solving integral equations

We can represent Volterra convolution integral equations of the first kind in the form:

$$intEq1 := \int_0^x k(x-t)v(t)dt = f(x)$$

where $v(x)$ is the unknown function to be determined, and $k(x)$(called the kernel) and $f(x)$ are known functions. There are different ways to solve such a problem. One commonly used method is Laplace Transform. Applying Laplace Transform to both sides of $intEq1$, and using the convolution property regarding Laplace Transform, then

$$K(s)V(s) = F(s)$$

where $V(s) = L[v(x)]$, $K(s) = L[k(x)]$, $F(s) = L[f(x)]$. Then

$$V(s) = \frac{F(s)}{K(s)}.$$

The conventional method to solve the problem is calculating the inverse of $V(s)$.

As listed above, both Sumudu and Laplace Transforms have the same or similar properties, therefore, Sumudu Transform can be used in the conventional method to solve such an equation. The drawback of this traditional method is to calculate the inverse transform, which is normally expensive and problematic. Can we avoid such an inverse transform? Yes, the following method can solve the problem without inverse transform:

- Transfer an integral equation to an algebraic equation by using Laplace Transform.
- Solve the algebraic equation generated by Laplace Transform.
- Calculate Sumudu Transform by using Laplace-Sumudu Duality.
- Calculate a_n, the general term of coefficient of the Sumudu Transform.
- Return the solution as the power series with $a_n/n!$ as the general term of coefficient.
- Optional, add the power series summation to a close form.

This method can be used to solve different kinds of integral equations, including the first, second and mixed kinds. Now, apply the method to $intEq1$, we have the following algorithm:

- Computing the Laplace Transforms $K(s) = L[k(x)]$, $F(s) = L[f(x)]$.
- Computing Laplace Transform $V(s) = \frac{F(s)}{K(s)}$.
- Computing the Sumudu Transform by Laplace-Sumudu Duality $S[v(x)] = V(1/s)/s$.
- Computing the nth coefficient a_n of $S[v(x)]$.
- Returning the solution as the power series with $a_n/n!$ as the general term of coefficient.
- Optional, adding the power series summation to a close form.

Since there are limitations in Maple systems, it is impossible to implement such an algorithm fully. For demonstration purpose, a simplified Maple procedure called $solveIntFirst$ can be implemented as the following:

$with(inttrans)$:
$with(genfunc)$:
$solveIntFirst := proc(v, k, f, x)$
  $local \quad F, K;$
  $F := laplace(f, x, s);$
  $F := simplify(subs(s = 1/t, F));$
  $K := laplace(k, x, s);$
  $K := simplify(subs(s = 1/t, K));$
  $K := simplify(F/(K * t));$
  $K := rgf\_expand(K, t, n);$
  $K := simplify(\frac{K * x^n}{n!});$
  $return \quad v(x) = sum(K, n = 0..\infty)$
$end \quad proc$

A simple way for testing the implementation is to compare with the built-in procedure called $intsolve(eq, y(x), method = Laplace)$, the results match except possibly 0 or a few $Dirac(x)$ functions, such kind of functions are 0 except a single point so we ignore in our solutions. The following are some sample test cases:

$eq := \int_0^x y(t)dt = 1;$
$> solveIntFirst(y, 1, 1, x);$
$Returns : y(x) = 0$
$> intsolve(eq, y(x), method = Laplace);$
$Returns : y(x) = Dirac(x)$

$eq := \int_0^x y(t)dt = \cos(x);$
$> solveIntFirst(y, 1, \cos(x), x);$
$Returns : y(x) = -\sin(x)$
$> intsolve(eq, y(x), method = Laplace);$
$Returns : y(x) = Dirac(x) - \sin(x)$

$eq := \int_0^x e^{x-t} y(t)dt = \cos(x);$
$> solveIntFirst(y, e^x, \cos(x), x);$
$Returns : y(x) = -\cos(x) - \sin(x)$
$> intsolve(eq, y(x), method = Laplace);$

$Returns: y(x) = Dirac(x) - \cos(x) - \sin(x)$

$eq := \int_0^x (x-t)^2 y(t)dt = \cos(x);$
$> solveIntFirst(y, x^2, \cos(x), x);$
$Returns: y(x) = \frac{\sin(x)}{2}$
$> intsolve(eq, y(x), method = Laplace);$
$Returns: y(x) = \frac{Dirac(2,x)}{2} - \frac{Dirac(x)}{2} + \frac{\sin(x)}{2}$

$eq := \int_0^x y(t)dt = \sin(x);$
$> solveIntFirst(y, 1, \sin(x), x);$
$Returns: y(x) = \cos(x)$
$> intsolve(eq, y(x), method = Laplace);$
$Returns: y(x) = \cos(x)$

$eq := \int_0^x e^{x-t} y(t)dt = \sin(x);$
$> solveIntFirst(y, e^x, \sin(x), x);$
$Returns: y(x) = \cos(x) - \sin(x)$
$> intsolve(eq, y(x), method = Laplace);$
$Returns: y(x) = \cos(x) - \sin(x)$

$eq := \int_0^x y(t)dt = e^x;$
$> solveIntFirst(y, 1, e^x, x);$
$Returns: y(x) = e^x$
$> intsolve(eq, y(x), method = Laplace);$
$Returns: y(x) = Dirac(x) + e^x$

$eq := \int_0^x \sin(x-t) y(t)dt = e^x;$
$> solveIntFirst(y, \sin(x), e^x, x);$
$Returns: y(x) = 2e^x$
$> intsolve(eq, y(x), method = Laplace);$
$Returns: y(x) = Dirac(1,x) + Dirac(x) + 2e^x$

$eq := \int_0^x \sin(x-t)^2 y(t)dt = e^x;$
$> solveIntFirst(y, \sin(x)^2, e^x, x);$
$Returns: y(x) = \frac{5e^x}{2}$
$> intsolve(eq, y(x), method = Laplace);$
$Returns: y(x) = \frac{Dirac(2,x)}{2} + \frac{Dirac(1,x)}{2} + \frac{5Dirac(x)}{2} + \frac{5e^x}{2}$

$eq := \int_0^x y(t)dt = \cosh(x);$
$> solveIntFirst(y, 1, \cosh(x), x);$
$Returns: y(x) = \frac{e^x}{2} - \frac{1}{2e^x} = \sinh(x)$
$> intsolve(eq, y(x), method = Laplace);$
$Returns: y(x) = Dirac(x) + \sinh(x)$

$eq := \int_0^x e^{x-t} y(t)dt = \cosh(x)$
$> solveIntFirst(y, e^x, \cosh(x), x);$
$Returns: y(x) = -\frac{1}{e^x}$
$> intsolve(eq, y(x), method = Laplace);$
$Returns: y(x) = Dirac(x) - e^{-x}$

$eq := \int_0^x y(t)dt = \sinh(x);$
$> solveIntFirst(y, 1, \sinh(x), x);$
$Returns : y(x) = \frac{e^x}{2} + \frac{1}{2e^x} = \cosh(x)$
$> intsolve(eq, y(x), method = Laplace);$
$Returns : y(x) = \cosh(x)$

$eq := \int_0^x e^{x-t}y(t)dt = \sinh(x);$
$> solveIntFirst(y, e^x, \sinh(x), x);$
$Returns : y(x) = \frac{1}{e^x}$
$> intsolve(eq, y(x), method = Laplace);$
$Returns : y(x) = e^{-x}$

The above solutions are in the original forms generated from $solveIntFirst$ with little change. These examples are just a few samples, and there are an infinite number of such equations that can be solved by the same way.

The calculation of the general term (nth term) of Taylor's coefficient is needed in the algorithm, Maple only works for rational functions and implemented by procedure rgf_expand, a potential problem was "inherited" from the procedure which assumes n is larger than the degree of the polynomial, therefore returns 0 for any polynomial functions (for example $rgf\_expand(x^{10} + x^6 + x + 2, x, n)$ returns 0). So the answer may be different from the exact one by a polynomial, however, tests show, in a lot of cases, it gives the exactly right answers.

Now, let's look at a different kind of integral equation. The Volterra convolution integral equations of the second kind can be represented as

$$intEq2 := f(x) + \int_0^x k(x-t)v(t)dt = v(x)$$

where $v(x)$ is the function to be determined, $k(x)$ (called the kernel) and $f(x)$ are given functions. Applying Laplace Transform to both sides of $intEq2$ and using the convolution property, we have

$$F(s) + K(s)V(s) = V(s)$$

where $F(s) = L[f(x)]$, $K(s) = L[k(x)]$, $V(s) = L[v(x)]$. So we have

$$V(s) = \frac{F(s)}{1 - K(s)}.$$

The conventional method to solve the problem is by calculating the inverse of $V(s)$.

Since Sumudu and Laplace Transforms have the same or similar properties, Sumudu Transform can also be used in the conventional method to solve such an equation. The drawback of this traditional method is to calculate the inverse transform, which is normally expensive and problematic. Can we avoid such an inverse transform? Yes, the following method can solve the problem without inverse transform:

- Calculate the Laplace Transforms $K(s) = L[k(x)]$, $F(s) = L[f(x)]$.
- Calculate $V(s) = \frac{F(s)}{1-K(s)}$.

- Calculate the Sumudu Transform by Laplace-Sumudu Duality $S[v(x)] = V(1/s)/s$.
- Calculate the nth coefficient a_n of $S[v(x)]$.
- Return the solution, the power series with $a_n/n!$ as the general term of coefficient.
- Optional, add the power series summation to a close form.

Since the limitations of current algebra systems, no one can implement such an algorithm completely and perfectly under the current systems, but a limited Maple procedure called *solveIntSecond* is implemented as follows:

$$
\begin{aligned}
&with(inttrans):\\
&with(genfunc):\\
&solveIntSecond := proc(v,k,f,x)\\
&\quad local \quad F,K;\\
&\quad F := laplace(f,x,s);\\
&\quad K := laplace(k,x,s);\\
&\quad K := simplify(F/(1-K));\\
&\quad K := simplify(subs(s=1/t,K)/t);\\
&\quad K := rgf\_expand(K,t,n);\\
&\quad K := simplify(K/n!);\\
&\quad return \quad v(x) = sum(K*x^n, n=0..\infty)\\
&end \quad proc
\end{aligned}
$$

Similarly, this implementation suffers the limitations and potential issues "inherited" from Maple as described in the above. However, there are still an infinite number of such kind of equations that can be solved by *solveIntSecond*, including but not limited to the following samples. One can compare and verify the results by comparing with solutions from $intsolve(eq, y(x), method = Laplace)$, and explore the equations of the following:

$eq := y(x) = 1 - \int_0^x (x-t)y(t)dt$
$> solveIntSecond(y, -x, 1, x);$
$Returns : y(x) = \cos(x)$
$> intsolve(eq, y(x), method = Laplace);$
$Returns : y(x) = \cos(x)$

$eq := y(x) = 1 + \int_0^x y(t)dt$
$> solveIntSecond(y, 1, 1, x);$
$Returns : y(x) = e^x$
$> intsolve(eq, y(x), method = Laplace);$
$Returns : y(x) = e^x$

$eq := y(x) = \cos(x) + \int_0^x (x-t)y(t)dt$
$> solveIntSecond(y, x, \cos(x), x);$
$Returns : y(x) = \frac{e^x}{4} + \frac{1}{4e^x} + \frac{\cos(x)}{2} = \frac{\cos(x)}{2} + \frac{\cosh(x)}{2}$
$> intsolve(eq, y(x), method = Laplace);$
$Returns : y(x) = \frac{\cos(x)}{2} + \frac{\cosh(x)}{2}$

$eq := y(x) = \cos(x) + \int_0^x y(t)dt$
$> solveIntSecond(y, 1, \cos(x), x);$
$Returns : y(x) = \frac{\cos(x)}{2} + \frac{\cosh(x)}{2} + \frac{\sin(x)}{2} + \frac{\sinh(x)}{2} = \frac{e^x}{2} + \frac{\cos(x)}{2} + \frac{\sin(x)}{2}$
$> intsolve(eq, y(x), method = Laplace);$
$Returns : y(x) = \frac{e^x}{2} + \frac{\cos(x)}{2} + \frac{\sin(x)}{2}$

$eq := y(x) = (\cos(x))^2 + \int_0^x (x-t)y(t)dt$
$> solveIntSecond(y, x, (\cos(x))^2, x);$
$Returns : y(x) = \frac{e^{2Ix}}{5} + \frac{1}{5e^{2Ix}} + \frac{3e^x}{10} + \frac{3}{10e^x} = \frac{4\cos(x)^2}{5} + \frac{3\cosh(x)}{5} - \frac{2}{5}$
$> intsolve(eq, y(x), method = Laplace);$
$Returns : y(x) = \frac{4\cos(x)^2}{5} + \frac{3\cosh(x)}{5} - \frac{2}{5}$

$eq := y(x) = \sin(x) + \int_0^x (x-t)y(t)dt$
$> solveIntSecond(y, x, \sin(x), x);$
$Returns : y(x) = \frac{e^x}{4} - \frac{1}{4e^x} + \frac{\sin(x)}{2} = \frac{\sin(x)}{2} + \frac{\sinh(x)}{2}$
$> intsolve(eq, y(x), method = Laplace);$
$Returns : y(x) = \frac{\sin(x)}{2} + \frac{\sinh(x)}{2}$

$eq := y(x) = \sin(x) + \int_0^x y(t)dt$
$> solveIntSecond(y, 1, \sin(x), x);$
$Returns : y(x) = -\frac{\cos(x)}{2} + \frac{\cosh(x)}{2} + \frac{\sin(x)}{2} + \frac{\sinh(x)}{2} = \frac{e^x}{2} - \frac{\cos(x)}{2} + \frac{\sin(x)}{2}$
$> intsolve(eq, y(x), method = Laplace);$
$Returns : y(x) = \frac{e^x}{2} - \frac{\cos(x)}{2} + \frac{\sin(x)}{2}$

$eq := y(x) = (\sin(x))^2 + \int_0^x (x-t)y(t)dt$
$> solveIntSecond(y, x, (\sin(x))^2, x);$
$Returns : y(x) = -\frac{e^{2Ix}}{5} - \frac{1}{5e^{2Ix}} + \frac{e^x}{5} + \frac{1}{5e^x} = -\frac{4\cos(x)^2}{5} + \frac{2\cosh(x)}{5} + \frac{2}{5}$
$> intsolve(eq, y(x), method = Laplace);$
$Returns : y(x) = -\frac{4\cos(x)^2}{5} + \frac{2\cosh(x)}{5} + \frac{2}{5}$

$eq := y(x) = \sin(x) + \int_0^x e^{(x-t)}y(t)dt$
$> solveIntSecond(y, e^x, \sin(x), x);$
$Returns : y(x) = \frac{-\cos(x)+\cosh(2x)+3\sin(x)+\sinh(2x)}{5} = \frac{e^{2x}}{5} - \frac{\cos(x)}{5} + \frac{3\sin(x)}{5}$
$> intsolve(eq, y(x), method = Laplace);$
$Returns : y(x) = \frac{e^{2x}}{5} - \frac{\cos(x)}{5} + \frac{3\sin(x)}{5}$

$eq := y(x) = \sinh(x) + \int_0^x (x-t)y(t)dt$
$> solveIntSecond(y, x, \sinh(x), x);$
$Returns : y(x) = \frac{e^x}{4} - \frac{1}{4e^x} + \frac{xe^x}{4} + \frac{x}{4e^x} = \frac{x\cosh(x)}{2} + \frac{\sinh(x)}{2}$
$> intsolve(eq, y(x), method = Laplace);$
$Returns : y(x) = \frac{x\cosh(x)}{2} + \frac{\sinh(x)}{2}$

$eq := y(x) = \sinh(x) - \int_0^x (x-t)y(t)dt$
$> solveIntSecond(y, -x, \sinh(x), x);$
$Returns : y(x) = \frac{\sin(x)}{2} + \frac{e^x}{4} - \frac{1}{4e^x} = \frac{\sinh(x)}{2} + \frac{\sin(x)}{2}$
$> intsolve(eq, y(x), method = Laplace);$
$Returns : y(x) = \frac{\sinh(x)}{2} + \frac{\sin(x)}{2}$

$eq := y(x) = \cosh(x) - \int_0^x (x-t)y(t)dt$
$> solveIntSecond(y, -x, \cosh(x), x);$
$Returns : y(x) = \frac{\cos(x)}{2} + \frac{e^x}{4} + \frac{1}{4e^x} = \frac{\cos(x)}{2} + \frac{\cosh(x)}{2}$
$> intsolve(eq, y(x), method = Laplace);$
$Returns : y(x) = \frac{\cos(x)}{2} + \frac{\cosh(x)}{2}$

We tested the samples above in the comparison with Maple built-in procedure $intsolve(eq, y(x), method = Laplace)$, it is no surprise that the results match exactly (the formats may be different). The above solutions are in the original forms generated directly from $solveIntSecond$ with little change.

7. Conclusion

Laplace Transform is a conventional transform which has been widely used for a long time. Sumudu Transform is recent, but it has many good properties for solving problems in sciences, technologies, pure and applied mathematics. Computational science and engineering is getting more and more important in problem solving, and Sumudu Transform is a powerful and effective new transform, in which various computational methods can be developed to solve different type of problems automatically. In this paper, the authors introduced Sumudu Transform in computational approach, with applications in automatic coefficient calculation, automatic proof of identities, automatic computation of generating functions, automatic solving of differential and integral equations, etc..

References

[1] S. K. Al-Omari, *The q-Sumudu transform and its certain properties in a generalized q-calculus theory,* Advances in Difference Equations, 2021, 2021, Article number: 10.

[2] S. Alfaqeih, G. Bakıcıerler and E. Misirli, *Conformable double Sumudu transform with applications,* Journal of Applied and Computational Mechanics, 2021, 7(2), 578–586.

[3] A. Atangana, *A new defy for iteration methods,* Journal of Applied Analysis and Computation, 2015, 5(3), 273–283.

[4] F. B. M. Belgacem, *Introducing and analysing deeper Sumudu properties,* Nonlinear Studies, 2006, 13(1), 23–41.

[5] F. B. M. Belgacem and A. A. Karaballi, *Sumudu transform fundamental properties investigations and applications,* Journal of Applied Mathematics and Stochastic Analysis, 2006, 2006, 1–23.

[6] C. Epstein and J. Schotland, *The Bad Truth about Laplace's Transform,* SIAM Review, 2008, 50(3), 504–552.

[7] A. Golmankhaneh and C. Tunç, *Sumudu transform in fractal calculus,* Applied Mathematics and Computation, 2019, 350, 386–401.

[8] M. Kapoor, *Sumudu transform for time fractional physical models an analytical aspect,* Journal of Applied Analysis and Computation, 2023, 13(3), 1255–1273.

[9] A. Kilicman and H. Hassan Eltayeb, *Some Remarks on the Sumudu and Laplace Transforms and Applications to Differential Equations,* ISRN Applied Mathematics, 2012.

[10] K. Kuhlman, *Review of inverse Laplace transform algorithms for Laplace-space numerical approaches,* Numerical Algorithms, 2013, 63, 339–355.

[11] M. S. Mechee and A. J. Naeemah, *A study of triple Sumudu transform for solving partial differential equations with some applications,* Multidisciplinary European Academic Journal, 2020, 2(2).

[12] M. Petkovsek, H. S. Wilfe and D. Zeilberger, $A = B$, A. K. Peters, Ltd, 1996.

[13] R. A. Ravenscroft, *Rational Generating Function Applications in Maple V: Mathematics and its Application,* R. Lopez (ed.), Birkhaser, 1994.

[14] R. A. Ravenscroft and E. A. Lamagna, *Symbolic summation with generating functions,* Proceedings of the International Symposium on Symbolic and Algebraic Computation, Association for Computing Machinery, 1989, 228–233.

[15] M. S. Rawashdeh and S. Maitama, *Solving nonlinear ordinary differential equations using the NDM*, Journal of Applied Analysis and Computation, 2015, 5(1), 77–88.

[16] D. V. Widder, *An Introduction to Transform Theory,* Academic Press, 1971.

[17] J. Zhang, *Sumudu Transform based coefficients calculation,* Nonlinear Studies, 2008, 15(4), 355–372.

[18] J. Zhang, R. Fan and F. Shen, *New method for the computation of generating functions with applications,* The 2021 International Conference on Computational Science and Computational Intelligence, IEEE, 2021, Las Vegas, USA.

[19] J. Zhang and F. Shen, *Sumudu transform for automatic mathematical proof and identity generation,* The 14th International Conference on Information Technology: New Generations, Springer, 2017, Las Vegas, USA.

[20] J. Zhang, F. Shen and Y. Waguespack, *Incorporating generating functions to computational science education*, The 2016 International Conference on Computational Science and Computational Intelligence, IEEE, 2016, Las Vegas, USA.

[21] J. Zhang, W. Zhu and F. Shen, *New techniques to solve differential equations automatically*, The 2017 International Conference on Computational Science and Computational Intelligence, IEEE, 2017, Las Vegas, USA.

[22] J. Zhang, W. Zhu and F. Shen, *New algorithms to solve integral equations automatically*, The 2019 International Conference on Computational Science and Computational Intelligence, IEEE, 2019, Las Vegas, USA.

[23] W. Zhao and S. Maitama, *Beyond Sumudu transform and natural transform: Transform properties and applications*, Journal of Applied Analysis and Computation, 2020, 10(4), 1223–1241.