DonnishJournals 2041-1189

Donnish Journal of Educational Research and Reviews.
Vol 1(2) pp. 041-046 December, 2014.
http://www.donnishjournals.org/djerr
Copyright © 2014 Donnish Journals

Original Research Paper

Laplace Transforms, Fast Fourier Transform, Fourier Series and Taylor Series with MAXIMA

Niyazi Ari and Savaş Tuylu*

Computer Science, Nigerian Turkish Nile University, Nigeria.

Accepted 15th December, 2014.

The methods of calculus lie at the heart of the physical sciences and engineering. Maxima can help you make faster progress, if you are just learning calculus. The examples in this research paper will offer an opportunity to see some Maxima tools in the context of simple examples, but you will likely be thinking about much harder problems you want to solve as you see these tools used here. This research paper includes Laplace Transforms, Fast Fourier Transform, Fourier Series and Taylor Series with MAXIMA

Keywords: Laplace Transform, Inverse Laplace Transform, Complex Fast Fourier Transform, Inverse Complex Fast Fourier Transform, Fourier Series, Taylor Series, Truncated Taylor, Maxima.

1. INTRODUCTION

Maxima is a system for the manipulation of symbolic and numerical expressions, including differentiation, integration, Taylor series, Laplace transforms, ordinary differential equations, systems of linear equations, polynomials, sets, lists, vectors, matrices, tensors, and more. Maxima yields high precision numeric results by using exact fractions, arbitrary precision integers, and variable precision floating point numbers. Maxima can plot functions and data in two and three dimensions.

Maxima source code can be compiled on many computer operating systems, including Windows, Linux, and MacOS X. The source code for all systems and precompiled binaries for Windows and Linux are available at the SourceForge file manager.

Maxima is a descendant of Macsyma, the legendary computer algebra system developed in the late 1960s at the Massachusetts Institute of Technology. It is the only system based on that effort still publicly available and with an active user community, thanks to its open source nature. Macsyma was revolutionary in its day, and many later systems, such as Maple and Mathematica, were inspired by it.

The Maxima branch of Macsyma was maintained by William Schelter from 1982 until he passed away in 2001. In 1998 he obtained permission to release the source code under the GNU General Public License (GPL). It was his efforts and skills that made the survival of Maxima possible.

MAXIMA has been constantly updated and used by researcher and engineers as well as by students.

2. BODE DIAGRAM

MAXIMA can evaluate a large variety of Laplace transform.

Corresponding Author: savastyludf@hotmail.com

Table 1.1: Description of the **laplace()** and inverse laplace transform **ilt()** functions.

Function	**Description**
laplace *(expr, t, s)*	Attempts to compute the Laplace transform of *expr* with respect to the variable *t* and transform parameter *s*.
ilt *(expr, s, t)*	Computes the inverse Laplace transform of *expr* with respect to *s* and parameter *t*. *expr* must be a ratio of polynomials whose denominator has only linear and quadratic factors. By using the functions laplace and ilt together with the solve or linsolve functions the user can solve a single differential or convolution integral equation or a set of them.

2.1. Examples

Example 2.1.1.

```
(%i1) eq:exp(-a*t^2);
      eq1:laplace(eq,t,s);
      ilt(eq1,s,t);
```

$$\text{(\%o1)}\quad \%e^{-a\,t^2}$$

$$\text{(\%o2)}\quad \frac{\sqrt{\pi}\,\%e^{\frac{s^2}{4\,a}}\left(1-\mathrm{erf}\left(\frac{s}{2\sqrt{a}}\right)\right)}{2\sqrt{a}}$$

$$\text{(\%o3)}\quad \mathrm{ilt}\left(\frac{\sqrt{\pi}\,\%e^{\frac{s^2}{4\,a}}\left(1-\mathrm{erf}\left(\frac{s}{2\sqrt{a}}\right)\right)}{2\sqrt{a}},s,\right.$$

Example 2.1.2.

```
(%i4) eq:1/sqrt(%pi*t^2);
      eq1:laplace(eq,t,s);
```

$$\text{(\%o4)}\quad \frac{1}{\sqrt{\pi}\,|t|}$$

$$\text{(\%o5)}\quad \frac{\mathrm{laplace}\left(\frac{1}{|t|},t,s\right)}{\sqrt{\pi}}$$

Example 2.1.3.

```
(%i6) eq:1/a*cos(a*t);
      eq1:laplace(eq,t,s);
      eqil:ilt(eq1,s,t);
```

$$\text{(\%o6)}\quad \frac{\cos(a\,t)}{a}$$

$$\text{(\%o7)}\quad \frac{s}{a\,(s^2+a^2)}$$

*Is a zero or nonzero?*nonzero

$$\text{(\%o8)}\quad \frac{\cos(a\,t)}{a}$$

Example 2.1.4.

```
(%i9) eq:'diff(f(x),x);
      eq1:laplace(eq,t,s);
      diff(diff(delta(t),t),t);
      laplace(%,t,s);
```

$$\text{(\%o9)}\quad \frac{d}{d\,x}\,f(x)$$

$$\text{(\%o10)}\quad \frac{\frac{d}{d\,x}\,f(x)}{s}$$

$$\text{(\%o11)}\quad \frac{d^2}{d\,t^2}\,\delta(t)$$

$$\text{(\%o12)}\quad -\left.\frac{d}{d\,t}\,\delta(t)\right|_{t=0}+s^2-\delta(0)\,s$$

3. FAST FOURIER TRANSFORM

The **fft** package comprises functions for numerical computation (not symbolic) of the fast fourier transform.

3.1. Examples

Example 3.1.1.

For real data

```
(%i6) load(fft)$
      fpprintprec:4$
      f:[-5,-1,-4,-3,3,4,1,5]$
      fi:inverse_fft(f);

      fii:fft(fi);
```

$$\text{(\%o9)}\quad [0.0,14.19\,\%i-5.879,1.0-1.0\,\%i,4.192\,\%i-10.12,-10.0,-4.192\,\%i-10.12,1.0\,\%i+1.0,-14.19\,\%i-5.879]$$

$$\text{(\%o10)}\quad [-5.0,-2.2204\,10^{-16}\,\%i-1.0,-4.0,-2.2204\,10^{-16}\,\%i-3.0,3.0\;2.2204\,10^{-16}\,\%i+4.0,1.0,2.2204\,10^{-16}\,\%i+5.0]$$

Table 2.1: Description of the fast fourier transfom and the inverse fast fourier transform functions.

Function	Description
fft *(x)*	Computes the complex fast Fourier transform. *x* is a list or array (named or unnamed) which contains the data to transform. The number of elements must be a power of 2. The elements must be literal numbers (integers, rationals, floats, or bigfloats) or symbolic constants, or expressions a + b*%i where a and b are literal numbers or symbolic constants. fft returns a new object of the same type as *x*, which is not modified. Results are always computed as floats or expressions a + b*%i where a and b are floats.
inverse_fft *(y)*	Computes the inverse complex fast Fourier transform. *y* is a list or array (named or unnamed) which contains the data to transform. The number of elements must be a power of 2. The elements must be literal numbers (integers, rationals, floats, or bigfloats) or symbolic constants, or expressions a + b*%i where a and b are literal numbers or symbolic constants. inverse_fft returns a new object of the same type as *y*, which is not modified. Results are always computed as floats or expressions a + b*%i where a and b are floats.

Example 3.1.2.

For complex data

```
(%i11) load(fft)$
       fpprintprec:4$
       f:[1,1+%i,1+%i,1,1,-1+%i,1-%i,-1]$
       fi:inverse_fft(f);

       fii:fft(fi);
(%o14) [2.0 %i +4.0 ,2.0 -2.828 %i ,2.0 ,-2.828 %i -2.0 ,4.0 -2.0 %i ,
2.828 %i +2.0 ,-2.0 ,2.828 %i -2.0 ]
 (%o15) [1.0 ,1.0 %i +1.0 ,1.0 %i +1.0 ,1.0 -1.1102 10^-16 %i ,1.0 ,1.0 %i
1.0 ,1.0 -1.0 %i ,1.1102 10^-16 %i -1.0 ]
```

Example 3.1.3.

Computation of sine and cosine coefficients

```
(%i16) load(fft)$
       fpprintprec:4$
       f:[1,2,3,4];
       n:length(f);
       x:make_array(any,n)$
       fillarray(x,f)$
       y:fft(x)$
       p:make_array(any,n/2+1)$
       q:make_array(any,n/2+1)$
       p[0]:realpart (y[0])$
       q[0]:0$
       for k:1 thru n/2-1 do
            (p[k]:realpart (y[k]+y[n-k]),
             q[k]:imagpart (y[k]-y[n-k]));
       p[n/2]:y[n/2]$
       q[n/2]:0$
       listarray(p);
       listarray(q);
       g(j):=sum(p[k]*cos(2*%pi*j*k/n)+p[k]*sin(2*%pi*j*k/n),k,0,n/2)$
       makelist(float(g(j)),j,0,n-1);
(%o18) [1,2,3,4]
 (%o19) 4
 (%o27) done
 (%o30) [2.5,-1.0,-0.5]
 (%o31) [0,-1.0,0]
 (%o33) [1.0,2.0,3.0,4.0]
```

Example 3.1.4.

```
(%i34) load(fft)$
       fpprintprec:4$
       f:[5,10,15,20,25,30,35,40];
       n:length(f);
       x:make_array(any,n)$
       fillarray(x,f)$
       y:fft(x)$
       p:make_array(any,n/2+1)$
       q:make_array(any,n/2+1)$
       p[0]:realpart (y[0])$
       q[0]:0$
       for k:1 thru n/2-1 do
            (p[k]:realpart (y[k]+y[n-k]),
             q[k]:imagpart (y[k]-y[n-k]));
       p[n/2]:y[n/2]$
       q[n/2]:0$
       listarray(p);
       listarray(q);
       g(j):=sum(p[k]*cos(2*%pi*j*k/n)+p[k]*sin(2*%pi*j*k/n),k,0,n/2)$
       makelist(float(g(j)),j,0,n-1);
(%o36) [5,10,15,20,25,30,35,40]
 (%o37) 8
 (%o45) done
 (%o48) [22.5,-5.0,-5.0,-5.0,-2.5]
 (%o49) [0,-12.07,-5.0,-2.071,0]
 (%o51) [5.0,12.93,25.0,22.93,25.0,27.07,25.0,37.07]
```

4. FOURIER SERIES

The **fourie** package comprises functions for the symbolic computation of Fourier series. There are functions in the **fourie** package to calculate Fourier integral coefficients and some functions for manipulation of expressions.

Table 4.1: The following table gives selected functions according to the fourier series:

Function	**Description**
fourier *(f, x, p)*	Returns a list of the Fourier coefficients of *f*(*x*) defined on the interval [-p, p].
foursimp *(l)*	Simplifies sin (n %pi) to 0 if sinnpiflag is true and cos (n %pi) to (-1)^n if cosnpiflag is true.
fourexpand *(l, x, p, limit)*	Constructs and returns the Fourier series from the list of Fourier coefficients *l* up through *limit* terms (*limit* may be inf). *x* and *p* have same meaning as in fourier.
fourcos *(f, x, p)*	Returns the Fourier cosine coefficients for *f*(*x*) defined on [0, *p*].
foursin *(f, x, p)*	Returns the Fourier sine coefficients for *f*(*x*) defined on [0, *p*].
totalfourier *(f, x, p)*	Returns fourexpand (foursimp (fourier (*f*, *x*, *p*)), *x*, *p*, 'inf).
fourint *(f, x)*	Constructs and returns a list of the Fourier integral coefficients of *f*(*x*) defined on [minf, inf].
fourintcos *(f, x)*	Returns the Fourier cosine integral coefficients for *f*(*x*) on [0,inf].
fourintsin *(f, x)*	Returns the Fourier sine integral coefficients for *f*(*x*) on [0, inf]

4.1. Examples

Example 4.1.1.

```
(%i1) load(fourie)$
      f:x^2+2$
      fourcos(f,x,1);
```

(%t3) $a_0=\frac{7}{3}$

(%t4) $a_n=2\left(\frac{3\sin(\pi n)}{\pi n}-\frac{2\sin(\pi n)}{\pi^3 n^3}+\frac{2\cos(\pi n}{\pi^2 n^2}\right.$

(%o4) [%t3,%t4]

Example 4.1.2.

```
(%i5) load(fourie)$
      f:x^2+2$
      fourier(f,x,1);
```

(%t7) $a_0=\frac{7}{3}$

(%t8) $a_n=2\left(\frac{3\sin(\pi n)}{\pi n}-\frac{2\sin(\pi n)}{\pi^3 n^3}+\frac{2\cos(\pi n}{\pi^2 n^2}\right.$

(%t9) $b_n=0$

(%o9) [%t7,%t8,%t9]

Example 4.1.3.

```
(%i10) load(fourie)$
       f:x^2+2$
       foursin(f,x,1);
```

(%t12) $b_n=2\left(\frac{2\sin(\pi n)}{\pi^2 n^2}-\frac{3\cos(\pi n)}{\pi n}+\frac{2\cos(\pi n)}{\pi^3 n^3}+\frac{2}{\pi n}-\frac{2}{\pi^3 n^3}\right)$

(%o12) [%t12]

Example 4.1.4.

```
(%i13) load(fourie)$
       f:x^2+2$
       totalfourier(f,x,1);
```

(%t15) $a_0=\frac{7}{3}$

(%t16) $a_n=2\left(\frac{3\sin(\pi n)}{\pi n}-\frac{2\sin(\pi n)}{\pi^3 n^3}+\frac{2\cos(\pi n}{\pi^2 n^2}\right.$

(%t17) $b_n=0$

(%t18) $a_0=\frac{7}{3}$

(%t19) $a_n=\frac{4(-1)^n}{\pi^2 n^2}$

(%t20) $b_n=0$

(%o20) $\dfrac{4\sum_{n=1}^{\infty}\frac{(-1)^n\cos(\pi n x)}{n^2}}{\pi^2}+\frac{7}{3}$

Example 4.1.5.

```
(%i21) load(fourie)$
       f:x^2+2$
       fourint(f,x);
```

(%t23) $a_z=\dfrac{2\left(\lim_{x\to\infty}\frac{x^2\sin(x z)}{z}+\frac{2\sin(x z)}{z}-\frac{2\sin(x z)}{z^3}+\frac{2x\cos(x z)}{z^2}\right)}{\pi}$

(%t24) $b_z=0$

(%o24) [%t23,%t24]

Example 4.1.6.

```
(%i25) load(fourie)$
       f:x^2+2$
       fourintcos(f,x);
```

(%t27) $a_z=\dfrac{2\left(\lim_{x\to\infty}\frac{x^2\sin(x z)}{z}+\frac{2\sin(x z)}{z}-\frac{2\sin(x z)}{z^3}+\frac{2x\cos(x z)}{z^2}\right)}{\pi}$

(%o27) [%t27]

Example 4.1.7.

```
(%i31) load(fourie)$
       f:x^2+2$
       fourintsin(f,x);
```

(%t33) $$b_z=\frac{2\left(\frac{\lim\limits_{x\to\infty}\frac{2\,x\sin(x\,z)}{z}+\frac{2\cos(x\,z)}{z^2}-x^2\cos(x\,z)-2\cos(x\,z)}{z}+\frac{2}{z}-\frac{2}{z^3}\right)}{\pi}$$

(%o33) [%t33]

5. TAYLOR SERIES

5.1. Examples

Example 5.1.1.

```
(%i1) eq:sqrt(sin(x)+a*x+1);
      ts:taylor(eq,x,0,3);
      %^2;
```

(%o1) $\sqrt{\sin(x)+a\,x+1}$

(%o2)/T/ $1+\frac{(a+1)\,x}{2}-\frac{(a^2+2\,a+1)\,x^2}{8}+\frac{(3\,a^3+9\,a^2+9\,a-1)\,x^3}{48}+..$

(%o3)/T/ $1+(a+1)\,x-\frac{x^3}{6}+...$

Example 5.1.2.

```
(%i4) eq:sqrt(x^3+1);
      ts:taylor(eq,x,0,5);
      %^2;
```

(%o4) $\sqrt{x^3+1}$

(%o5)/T/ $1+\frac{x^3}{2}+...$

(%o6)/T/ $1+x^3+...$

Example 5.1.3.

```
(%i7) eq:sin(x+y);
      ts:taylor(eq,x,0,5);
      %^2;
```

(%o7) $\sin(y+x)$

(%o8)/T/ $\sin(y)+\cos(y)\,x-\frac{\sin(y)\,x^2}{2}-\frac{\cos(y)\,x^3}{6}+\frac{\sin(y)\,x^4}{24}+\frac{\cos(y)\,x^5}{120}+..$

(%o9)/T/ $\sin(y)^2+2\cos(y)\sin(y)\,x+\left(-\sin(y)^2+\cos(y)^2\right)x^2-\frac{4\cos(y)\sin(y)\,x^3}{3}+\frac{(\sin(y)^2-\cos(y)^2)\,x^4}{3}+\frac{4\cos(y)\sin(y)\,x^5}{15}+...$

Example 5.1.4.

```
(%i10) eq:1/sin(x+y);
       ts:taylor(eq,x,0,5);
       %^2;
```

(%o10) $\frac{1}{\sin(y+x)}$

(%o11)/T/ $\frac{1}{\sin(y)}-\frac{\cos(y)\,x}{\sin(y)^2}+\frac{(\sin(y)^2+2\cos(y)^2)\,x^2}{2\sin(y)^3}-\frac{(5\cos(y)\sin(y)^2+6\cos(y)^3)\,x^3}{6\sin(y)^4}+\frac{(5\sin(y)^4+28\cos(y)^2\sin(y)^2+24\cos(y)^4)\,x^4}{24\sin(y)^5}-\frac{(61\cos(y)\sin(y)^4+180\cos(y)^3\sin(y)^2+120\cos(y)^5)\,x^5}{120\sin(y)^6}+...$

(%o12)/T/ $\frac{1}{\sin(y)^2}-\frac{2\cos(y)\,x}{\sin(y)^3}+\frac{(\sin(y)^2+3\cos(y)^2)\,x^2}{\sin(y)^4}-\frac{(8\cos(y)\sin(y)^2+12\cos(y)^3)\,x^3}{3\sin(y)^5}+\frac{(2\sin(y)^4+15\cos(y)^2\sin(y)^2+15\cos(y)^4)\,x^4}{3\sin(y)^6}-\frac{(34\cos(y)\sin(y)^4+120\cos(y)^3\sin(y)^2+90\cos(y)^5)\,x^5}{15\sin(y)^7}+...$

Example 5.1.5.

```
(%i13) eq:sin(x)^2;
       ts:taylor(eq,x,0,5);
       %^2;
```

(%o13) $\sin(x)^2$

(%o14)/T/ $x^2-\frac{x^4}{3}+...$

(%o15)/T/ $x^4-\frac{2\,x^6}{3}+...$

Example 5.1.5.

```
(%i16) eq:tan(x);
       ts:taylor(eq,x,0,5);
       %^2;
```

(%o16) $\tan(x)$

(%o17)/T/ $x+\frac{x^3}{3}+\frac{2\,x^5}{15}+...$

(%o18)/T/ $x^2+\frac{2\,x^4}{3}+\frac{17\,x^6}{45}+...$

Table 5.1: Description of the Taylor Series

Function	Description
taylor *(expr, x, a, n)* **taylor** *(expr, [x_1, x_2, ...], a, n)* **taylor** *(expr, [x, a, n, 'asymp])* **taylor** *(expr, [x_1, x_2, …], [a_1, a_2, …], [n_1, n_2, …])* **taylor** *(expr, [x_1, a_1, n_1], [x_2, a_2, n_2], …)*	taylor (*expr*, *x*, *a*, *n*) expands the expression *expr* in a truncated Taylor or Laurent series in the variable *x* around the point *a*, containing terms through (*x* - *a*)^*n*. If *expr* is of the form *f*(*x*)/*g*(*x*) and *g*(*x*) has no terms up to degree *n* then taylor attempts to expand *g*(*x*) up to degree 2 *n*. If there are still no nonzero terms, taylor doubles the degree of the expansion of *g*(*x*) so long as the degree of the expansion is less than or equal to *n* 2^taylordepth. taylor (*expr*, [*x_1*, *x_2*, ...], *a*, *n*) returns a truncated power series of degree *n* in all variables *x_1*, *x_2*, … about the point (*a*, *a*, ...). taylor (*expr*, [*x_1*, *a_1*, *n_1*], [*x_2*, *a_2*, *n_2*], ...) returns a truncated power series in the variables *x_1*, *x_2*, … about the point (*a_1*, *a_2*, ...), truncated at *n_1*, *n_2*, … taylor (*expr*, [*x_1*, *x_2*, ...], [*a_1*, *a_2*, ...], [*n_1*, *n_2*, ...]) returns a truncated power series in the variables *x_1*, *x_2*, … about the point (*a_1*, *a_2*, ...), truncated at *n_1*, *n_2*, … taylor (*expr*, [*x*, *a*, *n*, 'asymp]) returns an expansion of *expr* in negative Powers of *x* - *a*. The highest order term is (*x* - *a*)^-*n*. When maxtayorder is true, then during algebraic manipulation of (truncated) Taylor series, taylor tries to retain as many terms as are known to be correct. When psexpand is true, an extended rational function expression is displayed fully expanded. The switch ratexpand has the same effect. When psexpand is false, a multivariate expression is displayed just as in the rational function package. When psexpand is multi, then terms with the same total degree in the variables are grouped together.

6. CONCLUSION

The research paper can apply each and every part of Laplace Transforms, Fast Fourier Transform, Fourier Series and Taylor Series help application of the physical sciences and engineering, make faster progress, and help to understand Laplace Transforms, Fast Fourier Transform, Fourier Series and Taylor Series. The paper particularity helps to understand parts of Calculus and is going to extend to other parts of the Calculus.

7. ACKNOWLEDGEMENTS

I would like to thank the Maxima developers, Nigerian Turkish Nile University, and Prof. Niyazi ARI for their friendly help**.**

REFERENCES

[1] Niyazi ARI, Lecture notes, University of Technology, Zurich, Switzerland.
[2] Niyazi ARI, Symbolic computation of electromagnetics with Maxima (2013).
[3] http://maxima.sourceforge.net/
[4] R. H. Rand, Introduction to Maxima,
[5] R. Dodier, Minimal Maxima, 2005
[6] https://www.ma.utexas.edu/maxima/maxima_19.html.
[7] N. Ari, G. Apaydin, Symbolic Computation Techniques for Electromagnetics with MAXIMA and MAPLE,Lambert Academic Publishing, 2009.