DonnishJournals 2041-1189

Donnish Journal of Educational Research and Reviews. Vol 1(1) pp. 001-006 November, 2014.
http://www.donnishjournals.org/djerr

Original Research Paper

Differentiation and Integration Using MAXIMA

Savaş Tuylu

Computer Science, Nigerian Turkish Nile University, Nigeria.

Accepted 25th October, 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 Differentiation and Integration. After examples of using each topic some exercises have been formulated.

Keywords: Differentiation, Integration, Polynomial, Rational Founction, Logarithmic Function, Trigonometric Function, 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.

Differentiation provides symbolic Differentiation with derivative, a basic introduction to Differentiation, some functions and variables Polynomials, Explicit, Rational, Logarthmics and Trigonometrics using Maxima, definitions of Maxima's functions, derivative problem solving.

Integration has devoted symbolic Integration with integrate, solving integration examples, Polynomials, Rational and Trigonometric Functions using Maxima.
MAXIMA has been constantly updated and used by researcher and engineers as well as by students.

2. DIFFERENTIATION

2.1. Polynomials

Polynomials, in Maxima comes either in General Form or as Canonical Rational Expressions (CRE) form. The latter is a standard form, and is used internally by operations such as factor, ratsimp, and so on.
The ''variables" of a CRE expression needn't be atomic. In fact, any subexpression whose main operator is not +— */or with integer power will be considered a 'vari¬able" of the expression (in CRE form) in which it occurs. For example the CRE

Corresponding Author: savastyludf@hotmail.com

variables of the expression X + SIN(X + 1) + 2 * SQRT(X) + 1 are X, SQRT(X), and SIN(X + 1). If the user does not specify an ordering of variables by using the RATVARS function, Maxima will choose an alphabetic one. In general, CRE's represent rational expressions, that is, ratios of polynomials, where the numerator and denominator have no common factors, and the denominator is positive.

2.1.1. Some Functions and Variables for Polynomials

Function: **fullratsubst** (< a >, < b >, < c>) is the same as '**ratsubst**' except that it calls itself recursively on its result until that result stops changing. This function is useful when the replacement expression and the replaced expression have one or more variables in common.
fullratsubst will also accept its arguments in the format of **lratsubst**. That is, the first argument may be a single substitution equation or a list of such equations, while the second argument is the expression being processed.

load ("**lrats**") loads **fullratsubst** and **lratsubst**.

```
(%i1)  load ("lrats")$

-->   * 'subst' can carry out multiple substitutions.
'lratsubst' is analogous to 'subst'

(%i2) subst ([a=b,c=d],a+c);

(%o2) d+b

(%i3) lratsubst ([a^2 = b, c^2 = d], (a + e)*c*(a + c));

(%o3) (d+ac)e+ad+bc

(%i4) lratsubst (a^2 = b, a^3);

(%o4) ab

-->   * If only one substitution is desired, then a single
          equation may be given as first argument.

-->   * 'fullratsubst' is equivalent to 'subst' except that
        it recurses until its result stops changing.

(%i5) ratsubst (b*a, a^3, a^4);

(%o5) a^2 b

(%i6) fullratsubst (b*a, a^3, a^4);

(%o6) a^2 b

-->   * 'fullratsubst' also accepts a list of equations or a
              Single equation as first argument.

(%i7) fullratsubst ([a^3 = b, b^3 = c, c^3 = a], a^3*b*c);

(%o7) b^2 c

(%i8) fullratsubst (a^4 = b*a, a^3);

(%o8) a^3

-->   * 'fullratsubst' may cause an indefinite recursion

(%i9) errcatch (fullratsubst (b*a^2, a^2, a^3));

(%o9) [ ]
```

- Option vriable: **ratdenomdivide**
- Default value: 'true'

```
(%i10) expr: (x^3 + x + 1)/(y^4 + 7);

(%o10) (x^3+x+1)/(y^4+7)

(%i11) ratdenomdivide: true$

(%i12) ratexpand (expr);

(%o12) x^3/(y^4+7)+x/(y^4+7)+1/(y^4+7)

(%i13) ratdenomdivide: false$

(%i14) ratexpand (expr);

(%o14) (x^3+x+1)/(y^4+7)
```

- Function: **ratdiff** (< expr >, < x >)

Differentiates the rational expression < expr > with respect to < x >. < expr > must be a ratio of polynomials or a polynomial in < x >. The argument < x > may be a variable or a subexpression of < expr >.
The result is equivalent to **diff**, although perhaps in a different form. '**ratdiff**' may be faster than **diff**, for rational expressions. **ratdiff** returns a canonical rational expression (CRE) if **expr** is a CRE. Otherwise, **ratdiff** returns a general expression. **ratdiff** considers only the dependence of < expr > on < x >, and ignores any dependencies established by **depends**.

```
(%i15) expr: (2*x^2 + 5*x - 10)/(x^6 + 5);

(%o15) (2 x^2+5 x-10)/(x^6+5)

(%i16) ratdiff (expr, x);

(%o16) (8 x^7+25 x^6-60 x^5-20 x-25)/(x^12+10 x^6+25)
```

2.1.2. Differentiation of Explicit Functions

Beginning with explicit functions of a single variable and giving a few examples of the use of Maxima's **diff** function.

2.1.3. All about diff

The command **diff** (**expr, var, num**) will differentiate the expression in slot one with respect to the variable entered in slot two, a number of times determined by a positive integer in slot three. Unless a dependency has been established, all parameters and "variables" in the expression are treated as constants when taking the derivative. Thus **diff** (expr, x, 2) will yield the second derivative of **expr** with respect to the variable x. The simple form **diff** (expr, var) is equivalent to **diff** (**expr, var, 1)** .
If the expression depends on more than one variable, we can use commands such as **diff** (**expr, x, 2, y, 1**) to find the result of taking the second derivative with respect to **x** (holding **y** fixed) followed by the first derivative with respect to **y** (holding **x** fixed).

Example 1. Calculate the derivative of x^n?

```
(%i1) diff(x^n,x);

(%o1) n x^(n-1)
```

Note that x_i is Maxima's way of "pretty printing" x **[1]**. We can use the **grind** function to display the output **o1** in the "non-

pretty print mode" (what would have been relumed if we had set the **display2d** switch to **false**).

```
(%i10) diff(x[1]^2+x[2]^2,x[1]);
(%o10) 2 x_1
(%i5) grind(%)$
2*x[1]$
(%i6) display2d$
```

*Note the dollar sign adds to the end of its output.

Finally, an example of using one invocation of **diff** to differentiate with respect to more than one variable:

```
(%i11) diff(x^2*y^3,x,1,y,2);
(%o11) 12 x y
```

2.2. DERIVATIVE OF SOME RATIONAL FUNCTIONS

2.2.1. The Total Differential

If you use the **diff** function without a symbol in slot two, Maxima returns the "total differential" of the expression in slot one, and by default assumes every parameter is a variable. If an expression contains a single parameter, say **x**, then **diff(expr)** will generate:

$$\mathrm{df(x)}=\left(\frac{\mathrm{df(x)}}{\mathrm{dx}}\right)\mathrm{dx} \quad (2.2.1.1)$$

In the first example below, we calculate the differential of the expression x^2, that is, the derivative of the expression (with respect to the variable x) multiplied by "the differential of the independent variable x", dx. Maxima uses **del(x)** for the differential of x, a small increment of x.

```
(%i1) diff(x^2);
(%o1) 2 x del(x)
```

If the expression contains two parameters x and y, then the total differential is equivalent to the Maxima expression.

diff(expr,x)*del(x) + diff(expr,y)*del(y),

which generates (using conventional notation):

$$\mathrm{df(x,y)}=\left(\frac{\partial f(\mathrm{x,y})}{\partial x^2}\right)_y dx+\left(\frac{\partial f(\mathrm{x,y})}{\partial y}\right)_x dy \quad (2.2.1.2)$$

Each additional parameter induces an additional term of the same form.

```
(%i1) diff(x^2*y^3);
(%o1) 3 x^2 y^2 del(y)+2 x y^3 del(x)
(%i2) diff(a*x^2*y^3);
(%o2) 3 a x^2 y^2 del(y)+2 a x y^3 del(x)+x^2 y^3 del(a)
```

We can use the **subst** function to replace, say **del (x)**, by anything else:

```
(%i3) subst(del(x)=dx,%o1);
(%o3) 3 x^2 y^2 del(y)+2 dx x y^3
(%i4) subst([del(x)=dx,del(y)=dy,del(a)=da],%o3);
(%o4) 2 dx x y^3+3 dy x^2 y^2
```

You can use the declare function to prevent some of the symbols from being treated as variables when calculating the total differential:

```
(%i1) declare(a,constant)$
(%i2) diff(a*x^3*y^3);
(%o2) 3 a x^3 y^2 del(y)+3 a x^2 y^3 del(x)
(%i3) declare([b,c],constant)$
(%i4) diff(a*x^3+b*x^2+c*x);
(%o4) (3 a x^2+2 b x+c) del(x)
(%i5) properties(a);
(%o5) [database info,kind(a,constant)]
(%i6) propvars(constant);
(%o6) [a,b,c]
(%i7) kill(a,b,c)$
(%i8) propvars(constant);
(%o8) []
```

We can "map" **diff** on to a list of functions of **x**, say, and divide by **del (x)**, to generate a list of derivatives, as in

```
(%i10) map('diff,[sin(x),cos(x),tan(x)])/del(x);
(%o10) [cos(x),-sin(x),sec(x)^2]

(%i11) map('diff,%o10)/del(x);
(%o11) [-sin(x),-cos(x),2 sec(x)^2 tan(x)]
```

2.2.2. Logarithmic Functions

The logarithm $\log_b x$ for a base b and a number x is defined to be the inverse function of taking b to the power x, i.e., b^x. Therefore, for any x and b,

$$x=\log_{b^x}(b^x) \quad (2.2.2.1)$$

or equivalently,

$$\mathrm{x}=\mathrm{b}^{\log_{b^x}} \quad (2.2.2.2)$$

Example 1. Find the derivative of $\mathrm{y}=\log\frac{1}{1-x}$?

Solution:

```
(%i1) diff(y=log(1/(1-x)));
(%o1) del(y)=del(x)/(1-x)
```

2.3. TRIGONOMETRIC FUNCTIONS

2.3.1 Some Functions and Variables for Trigonometric

These operate more or less as you would expect. The following functions are defined by default:

Table: Some Trigonometric Functions

Command	Description
acos (< x >)	Arc Cosine
acosh (< x >)	Hyperbolic Arc Cosine
cot (< x >)	Arc Cotangent
acoth (< x >)	Hyperbolic Arc Cotangent
atan (< x >)	Arc Tangent

- **Package: atrig1**

The **atrig1** package contains several additional simplification rules for inverse trigonometric functions. Together with rules already known to Maxima, the following angles are fully implemented: **0**, %**pi/6**, %**pi/4**, %**pi/3**, and **%**pi/2. Corresponding angles in the other three quadrants are also available. Do **load(atrig1)**; to use them.

There are a couple wrinkles worth noting - by default, Maxima will not simplify expressions which numerically are nice fractions of π, so there exists a package which may be loaded to allow this called **atrig1**. Maxima is aware of the Half Angle relations, but by default will not use them. There is a variable which can be set called **halfangles**, and when that is set to true the Half Angle definitions will be used.

```
(%i1) acos(1/sqrt(2));
(%o1) π/4
```

There are a few global variables you can set which will change how Maxima handles trig expressions:

- **Triginverses**

This can be set to one of three values: ALL, TRUE, or FALSE. The default is ALL

- **ALL:** When set to ALL, both arctfun (tfun (x)) and fun(arctfun(x)) are evaluated to x.
- **TRUE:** When set to TRUE, the arctfun(tfun(x)) simplification is turned off.
- **FALSE:** When set to FALSE, both simplifications are turned off.

- **Trigsign**

Can be set to TRUE or FALSE. The default is TRUE. If TRUE, for example, sin(-x)

2.3.2. Differentiaton

To differentiate an expression, use the **diff** command. **diff(expr,var)** differentiates an expression with respect to the variable **var**.

```
(%i1) sin(x)*cos(x);
(%o1) cos(x)sin(x)
```

To take a second order derivative, use **diff**(expr,var,2).

```
(%i2) diff(sin(x)*cos(x),x,2);
(%o2) -4 cos(x)sin(x)
```

Differentiation, unlike integration, can be handled in a fairly general way by com¬puter algebra. As a result, you will be able to take derivatives in most cases. I will show an example here:

```
(%i3) diff((x^3+5)^6*(x^11-3*x-2)^21,x);
(%o3) 18 x^2 (x^3+5)^5 (x^11-3 x-2)^21 +21 (x^3+5)^6 (11 x^10-3) (x^11-3 x-2)^20
```

2.3.3. Controlling the Form of a Derivatives with gradef

We can use **gradef** to select one from among a number of alternative ways of writing the result of differentiation. The large number of "trigonometric identities" means that any given expression containing trig functions can be written in terms of a different set of trig functions. As an example, consider trig identities which express **sin(x)**, **cos(x)**, and **sec2(x)** : in terms of **tan(x)** and **tan(x/2)**:

$$\sec^2(x) = 1 + \tan^2(x) \quad (2.3.3.1),$$

$$\sin x = 2\tan(x/2)/(1+\tan^2(x/2)) \quad (2.3.3.2),$$

$$\cos x = (1-\tan^2(x/2))/(1+\tan^2(x/2)) \quad (2.3.3.3).$$

The default Maxima result for the first derivatives of sin x, cos x, and tan x is:

```
(%i1) map('diff,[sin(x),cos(x),tan(x)])/del(x);
(%o1) [cos(x),-sin(x),sec(x)^2]
```

Before using **gradef** to alter the return value of **diff** for these three functions, let's check that Maxima agrees with the trig "identities" displayed above. Variable amounts of expression simplification are needed to get what we want.

```
(%i2) trigsimp(1+tan(x)^2);
(%o2) 1/cos(x)^2
```

Recall that **trigsimp** will convert **tan**, **sec**, etc to sin and cos of the same argument. Recall also that **trigreduce** will convert an expression containing **cos (x/2)** and **sin (x/2)** into an expression containing **cos (x)** and **sin (x)**. Finally, remembering that six = 1/cosx, we see that Maxima has "passed the examination".

Now let's use **gradef** to express the derivatives in terms of **tan x** and **tan(x/2)**:

```
(%i1) gradef(tan(x),1+tan(x)^2);
(%o1) tan(x)
(%i2) gradef(sin(x),(1-tan(x/2)^2)/(1+tan(x/2)^2));
(%o2) sin(x)
(%i3) gradef(cos(x),-2*tan(x/2)/(1+tan(x/2)^2));
(%o3) cos(x)
```

Here we check the behavior:

```
(%i4) map('diff,[sin(x),cos(x),tan(x)])/del(x);
```

$$(\%o4)\quad \left[\frac{1-\tan\left(\frac{x}{2}\right)^2}{\tan\left(\frac{x}{2}\right)^2+1},-\frac{2\tan\left(\frac{x}{2}\right)}{\tan\left(\frac{x}{2}\right)^2+1},\tan(x)^2+1\right]$$

3. INTEGRATION

3.1. Symbolic Integration with integrate

The Maxima manual entry for **integrate** includes:

Function: **integrate(expr, var)**
Function**: integrate(expr, var, a, b)**

Attempts to symbolically compute the integral of **expr** with respect to **var**. **Integrate(expr, var)** is an indefinite integral, while **integrate(expr, var, a, b**) is a definite integral, with limits of integration a and b. The integral is returned if integrate succeeds. Otherwise the return value is the noun form of the integral (the quoted operator **integrate**) or an expression containing one or more noun forms. The noun form of the **integrate** is displayed with an integral sign if **display2d** is set to **true** (which is the default).

In some circumstances, it is useful to construct a noun form by hand, by quoting **integrate** with a single quote, e.g., **'integrate(expr, var)**. For example, the integral may depend on some parameters which are not yet computed. The noun may be applied to its arguments by **ev(iexp**, nouns) where **iexp** is the noun form of interest.

The Maxima function **integrate** is defined by the lisp function **integrate** in the file /src/simp.lisp. The indefinite integral invocation, **integrate(expr,var)**, results in a call to the lisp function **sinint**, defined in src/sin.lisp, unless the flag **risch** is present, in which case the lisp function **rischint**, defined in src/risch.lisp, is called. The definite integral invocation, **integrate(expr,var,a,b)**, causes a call to the lisp function **defint**, defined in src/defint.lisp.

The lisp function **defint** is available as the Maxima function **defint** and can be used to bypass **integrate** for a definite integral. To integrate a Maxima function **f (x)**, insert **f (x)** in the **expr** slot. **integrate** does not respect implicit dependencies established by the **depends** function. **integrate** may need to know some property of the parameters in the integrand.

integrate will first consult the **assume** database, and, if the variable of interest is not there, **integrate** will ask the user. Depending on the question, suitable responses are **yes;** or **no;**, or **pos;**, **zero;**, or **neg;**. Thus, the user can use the **assume** function to avoid all or some questions.

3.1.1. Integration Examples and also defint and Idefint

Example 1. Determine the indefinite integrate, $\int \sin^5 x dx$?

Solution:

```
(%i1) integrate(sin(x)^5,x);
```

$$(\%o1)\quad -\frac{\cos(x)^5}{5}+\frac{2\cos(x)^3}{3}-\cos(x)$$

```
(%i2) (diff(%,x),trigsimp(%%));
(%o2) sin(x)^5
```

Notice that the indefinite integral returned by **integrate** does not include the arbitrary constant of integration which can always be added. If the returned integral is correct (up to an arbitrary constant), then the first derivative of the returned indefinite integral should be the original integrand, although we may have to simplify the result manually (as we had to do above).

The above definite integral can be related to the "area under a curve" and is the more accessible concept, while the integral is simply a function whose first derivative is the original integrand. Instead of using **integrate** for a definite integral, you can try **Idefint** (think Limit definite integral), which may provide an alternative form of the answer. From the Maxima manual:

Function: **Idefint(expr, x, a, b)**

Attempts to compute the definite integral of **expr** by using **limit** to evaluate the indefinite integral of **expr** with respect to x at the upper limit b and at the lower limit a. If it fails to compute the definite integral, **Idefint** returns an expression containing limits as noun forms. **Idefint** is not called from **integrate**, so executing **defint(expr, x, a, b)** may yield a different result than integrate(expr, x, a, b)**. Idefint** always uses the same method to evaluate the definite integral, while **integrate** may employ various investigative approaches and may recognize some special cases.

Example 2. An example of a definite integral over an infinite range, $\int_{-\infty}^{\infty} x^2 e^{-x^2} dx$?

Solution:

```
(%i3) integrate(x^3*exp(-x^2),x,minf,inf);
(%o3) 0
```

3.2. Some examples of Polynomials and Rational Functions

Example 1. Evaluate the integral, $\int x^2 dx$?

Solution:

```
(%i1) integrate(x^2,x);
```

$$(\%o1)\quad \frac{x^3}{3}$$

Example 2. Find the antiderivative, $\int \frac{1}{x^2} dx$?

Solution:

```
(%i2) integrate(1/x^2,x);
(%o2) -1/x
```

3.3. Some examples of Trigonometric Functions

Unlike differentiation, integration cannot be readily expressed in a general way. Maxima is quite capable when it comes to such problems, although like all computer algebra systems it has its limits.

Example 1.

```
(%i3) integrate(log(x)^2/x,x);
(%o3) log(x)^3/3
```

Example 2.

```
(%i4) integrate(sin(a*x)/x,x);
(%o4) -(%i gamma_incomplete(0,%i a x)-%i gamma_incomplete(0,-%i a x))/2
```

4. CONCLUSION

Research paper can apply each and every part of Differentiation and Integration, help application of the physical sciences and engineering, make faster progress, and help to understand Differentiation and Integration faster. The paper particularity help to understand parts of Calculus and is going to extend to other parts of the Calculus.

5. 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] Edwin L. Woollett, Maxima by Example.
[4] http://euler.rene-grothmann.de/reference/maximacore.html.
[5] https://www.ma.utexas.edu/maxima/maxima_19.html.