Some Useful Maple Commands

`diff`: Compute symbolic derivatives

> ```
> ex1 := exp( sin(4*x) ) + ln( x^2 + sqrt(x)
> );
> ```

$$ex1 := \mathbf{e}^{\sin(4\,x)} + \ln(x^2 + \sqrt{x})$$

> ```
> diff( ex1, x );
> ```

$$4 \cos(4\,x)\,\mathbf{e}^{\sin(4\,x)} + \frac{2\,x + \dfrac{\dfrac{1}{2}}{\sqrt{x}}}{x^2 + \sqrt{x}}$$

> ```
> diff( ex1, x$2 );
> ```

$$-16 \sin(4\,x)\,\mathbf{e}^{\sin(4\,x)} + 16 \cos(4\,x)^2\,\mathbf{e}^{\sin(4\,x)} + \frac{2 - \dfrac{1}{4}\dfrac{1}{x^{(3/2)}}}{x^2 + \sqrt{x}}$$

$$- \frac{\left(2\,x + \dfrac{\dfrac{1}{2}}{\sqrt{x}} \right)^2}{(x^2 + \sqrt{x})^2}$$

```
> diff( diff( cos(x/y) , x ) , y );
```

$$\frac{\cos\left(\frac{x}{y}\right)x}{y^3} + \frac{\sin\left(\frac{x}{y}\right)}{y^2}$$

int: Compute symbolic integrals (anti-derivatives and definite integrals)

```
> ex2 := 1 / sqrt( x^2 - a^2 );
```

$$ex2 := \frac{1}{\sqrt{x^2 - a^2}}$$

```
> int( ex2, x );
```

$$\ln(x + \sqrt{x^2 - a^2})$$

A definite integral

```
> int( subs(a=0,ex2), x=1..2 );
```

$$-\frac{1}{3}\sqrt{-3}$$

Don't expect miracles!!

```
> int( ex1, x );
```

$$\int e^{\sin(4\,x)} + \ln(x^2 + \sqrt{x})\,dx$$

Multi-dimensional integrals are often straightforward:

```
> ex3 := (x^3 - y^3) / (x^2 + y^2);
```

$$ex3 := \frac{x^3 - y^3}{x^2 + y^2}$$

```
> int(int(ex3,x),y);
```

$$\frac{1}{6}x^2\,y - \frac{1}{6}y^3\,\ln(x^2+y^2) + \frac{1}{9}y^3 + \frac{1}{3}x^3\,\arctan\!\left(\frac{y}{x}\right) - \frac{1}{3}y^3\,\arctan\!\left(\frac{x}{y}\right)$$

$$-\frac{1}{6}x\,y^2 - \frac{1}{3}x^3\,\ln\!\left(\frac{x}{y}\right) + \frac{1}{6}x^3\,\ln\!\left(\frac{x^2}{y^2}+1\right)$$

`series` and `taylor`: Compute power series expansions

```
> series(exp(x),x=0);
```

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

```
> series(exp(x),x=0,11);
```

$$1 + x + \frac{1}{2}x^2 + \frac{1}{6}x^3 + \frac{1}{24}x^4 + \frac{1}{120}x^5 + \frac{1}{720}x^6 + \frac{1}{5040}x^7 + \frac{1}{40320}x^8$$

$$+ \frac{1}{362880}x^9 + \frac{1}{3628800}x^{10} + O(x^{11})$$

```
> taylor(exp(x),x=0);
```

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

```
> Order := 8;
```

$$Order := 8$$

```
> taylor(exp(x),x=0);
```

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

```
> Order := 6:
```

```
> taylor((1 + x)^(-1),x=0,7);
```

$$1 - x + x^2 - x^3 + x^4 - x^5 + x^6 + O(x^7)$$

```
> convert(%,'polynom');
```
$$1 - x + x^2 - x^3 + x^4 - x^5 + x^6$$

```
> coeffs(%,'x');
```
$$1, -1, -1, 1, 1, -1, 1$$

solve: Solve equations, including linear systems

```
> eq1 := x + y + z = 6;   eq2 := 2*x + y + z =
  2;   eq3 := x + y + 3 *z = 3;
```
$$eq1 := x + y + z = 6$$
$$eq2 := 2\,x + y + z = 2$$
$$eq3 := x + y + 3\,z = 3$$

```
> solve( {eq1,eq2,eq3} );
```
$$\{z = \frac{-3}{2}, y = \frac{23}{2}, x = -4\}$$

```
> solve( {eq1,eq2,eq3} , {x,y,z} );
```
$$\{x = -4, y = \frac{23}{2}, z = \frac{-3}{2}\}$$

```
> eq[1] := c[1] + c[2] + c[3] = 6:
> eq[2] := 2*c[1] + c[2] + c[3] = 2:
> eq[3]:= c[1] + c[2] + 3*c[3] = 3:
> eq[1]; eq[2]; eq[3];
```

$$c_1 + c_2 + c_3 = 6$$

$$2\,c_1 + c_2 + c_3 = 2$$

$$c_1 + c_2 + 3\,c_3 = 3$$

```
> solve( {eq[1],eq[2],eq[3]},
>        { c[1], c[2], c[3]} );
```

$$\{c_3 = \frac{-3}{2}, c_2 = \frac{23}{2}, c_1 = -4\}$$

```
> solve( {eq[1],eq[2],eq[3]} );
```

$$\{c_3 = \frac{-3}{2}, c_2 = \frac{23}{2}, c_1 = -4\}$$

Common Mistakes and Helpful Hints

(0) Not terminating statement with ';' or ':' *(but note that Maple V.5 and later versions now warn of "premature end of input")*

```
> a := int(x^2,x=0..1)
> ;
```

$$a := \frac{1}{3}$$

(1) Using '=' rather than ':=' for assignment

```
> a = 2;
```

$$\frac{1}{3} = 2$$

```
> a;
```

$$\frac{1}{3}$$

```
> a := 2;
```

$$a := 2$$

```
> a;
```

$$2$$

(2) Using quotes (') rather than double quotes (") to delimit strings

```
> string1 := "This is a Maple string";
                string1 := "This is a Maple string"
```

```
> string2 := 'This is not a Maple string';
Error, missing operator or ';'
```

(3) Unique syntax for programming constructs

```
> for i from 1 to 2 do
>    print(i);
> od;
                               1
                               2
```

(4) Use 'eval' or 'op' command to display definitions

```
> mysum := proc(x,y) x + y end:
```

```
> eval(mysum);
                    proc(x, y) x + y end proc
```

```
> op(mysum);
                    proc(x, y) x + y end proc
```

(5) Use 'interface(verboseproc=2)' to display system proc. defns.

```
> interface(verboseproc=2);
```

```
> eval(nops);
```
$$\textbf{proc}()\ \textbf{option}\ \textit{builtin};\ 203\ \textbf{end proc}$$

```
> eval(sin);
```
$\textbf{proc}(x::\textit{algebraic})$

$\textbf{local}\ n, t, \textit{pull\_out}, \textit{keep\_in};$

$\textbf{option}$ *'Copyright (c) 1992 by the University of Waterloo. All\ri*

ghts reserved.' ;

    $\textbf{if}\ \text{nargs} \neq 1\ \textbf{then}$

        $\textbf{error}$ "expecting 1 argument, got %1", nargs

    $\textbf{elif}\ \text{type}(x, \text{'complex}(\textit{float})\text{'})\ \textbf{then}\ \text{evalf('sin'}(x))$

    $\textbf{elif}\ \text{type}(x, \text{'}\infty\text{'})\ \textbf{then}$

        $\textbf{if}\ \text{type}(\Re(x), \text{'}\infty\text{'})\ \textbf{then}\ x{*}\textit{undefined}$

        $\textbf{elif}\ \text{type}(x, \text{'}\textit{imaginary}\text{'})\ \textbf{then}\ x$

        $\textbf{else}\ \infty + \infty{*}I$

        $\textbf{end if}$

    $\textbf{elif}\ \text{type}(x, \text{'}\textit{undefined}\text{'})\ \textbf{then}\ x{*}\textit{undefined}$

    $\textbf{elif}$

    $\text{type}(x, \text{'}\textit{SymbolicInfinity}\text{'})\ \textbf{and}\ \text{traperror}(\text{is}(x, \textit{real})) = \textit{true}$

then *undefined*

elif type(*x*, *'imaginary'*) **or** type(*x*, '*') **and**

member(*true*, map(*type*, { op(*x*) }, *'imaginary'*)) **then**

$I*\sinh(-I*x)$

elif type(*x*, *'complex(numeric)'*) **then**

if csgn(*x*) < 0 **then** −sin(−*x*) **else** *'sin'*(*x*) **end if**

elif type(*x*, '*') **and** type(op(1, *x*), *'complex(numeric)'*) **and**

csgn(op(1, *x*)) < 0 **then** −sin(−*x*)

elif type(*x*, '*') **and** type(*x*, *''&*'(rational*, identical(π)))'*)

then

$t := \text{op}(1, x);$

if *t* < 1 / 2 **then** *'sin'*(*x*)

elif *t* < 1 **then** sin((1 − *t*)*π)

elif *t* < 2 **then** −sin((2 − *t*)*π)

else sin((*t* − 2*iquo(trunc(*t*), 2))*π)

end if

elif type(*x*, '*') **and**

select(*type*, [op(*x*)], *'specfunc'*(*'anything'*, *'csgn'*)) ≠ [ ] **then**

pull_out, *keep_in* := selectremove(*type*, *x*,

'specfunc'(*'anything'*, *'csgn'*));

pull_out*sin(*keep_in*)

elif type(*x*, *'specfunc'*(*'anything'*, *'csgn'*)) **then** *x**sin(1)

elif type(*x*, '+') **and** traperror(sign(*x*)) = -1 **then** −sin(−*x*)

elif type(x, '$+$') **and** has(x, π) **then**

    $t :=$ map(**proc**(x)

                if type(x / π, '*rational*') **then** x / π **end if**

        end proc, $\{\,$op(x)$\,\}$);

    if nops(t)$= 1$ **then**

        $t :=$ op(t);

        if $t < 0$ **then** $\sin(x - 2*\pi*\mathrm{trunc}(1/2*t) + 2*\pi)$

        elif $t < 1/2$ **then** $\sin(x) :=$ '$\sin$'(x)

        elif $t < 1$ **then** $\cos(x - 1/2*\pi)$

        elif $t < 2$ **then** $-\sin(x - \pi)$

        else $\sin(x - 2*\pi*\mathrm{trunc}(1/2*t))$

        end if

    else $\sin(x) :=$ '$\sin$'(x)

    end if

elif type(x, '$*$') **and** member(π, [op(x)], 'n') **and** $\Im(x) = 0$ **then**

    $t :=$ subsop($n = 1, x$);

    $n :=$ frac(t);

    if $n = 0$ **then** 0

    elif frac($1/2*t - 1/4$)$= 0$ **then** 1

    elif frac($1/2*t + 1/4$)$= 0$ **then** -1

    elif frac($t - 1/2$)$= 0$ **then** $(-1)^{\wedge}(t - 1/2)$

    else $\sin(x) :=$ '$\sin$'(x)

```
        end if
    elif type(x, 'function') and nops(x) = 1 then
        n := op(0, x);
        t := op(1, x);
        if n = 'arcsin' then t
        elif n = 'arccos' then sqrt(1 - t^2)
        elif n = 'arctan' then t / sqrt(1 + t^2)
        elif n = 'arccsc' then 1 / t
        elif n = 'arcsec' then sqrt(1 - 1 / t^2)
        elif n = 'arccot' then 1 / sqrt(1 + t^2)
        else sin(x) := 'sin'(x)
        end if
    elif type(x, 'specfunc'('anything', 'JacobiAM')) then
        JacobiSN(op(x))
    elif type(x, 'arctan(algebraic, algebraic)') then
        op(1, x) / sqrt(op(1, x)^2 + op(2, x)^2)
    else sin(x) := 'sin'(x)
    end if
end proc
```