Maxima Notes

Created 2011-02-05 Modified

Maxima programming examples CAS Maxima Manual Examples
Using Maxima as a calculator
Issues

An open source alternative to Mathematica - to a large extent.
Mathematica vs Lisp Syntax; Maxima is written in Lisp.

Not case sensitive.
$ suppresses printing.
Each entry generates a variable that can be referred to. e.g. (C4) entry as EV(c4,x=2.6);
kill(name); to remove definition.

Comparison: finicky and flaky. Use the KISS principle, space between arguments seems to help.

```
if 4 = 3 then print("1");
```

false

```
if 4 = 4 then print("1");
```

1

```
if 4 <= 4 then print("1");
```

1

`if (4 = 4) then print("1");` - failed and subsequently ran? Who is home? A temperamental compiler is a huge negative on productivity. I dislike having to debug code for a simple task, over an hour, only to find the same code starts working.

```
if ( 4 = 4 ) and ( 1 = 1 ) then print("2");
```

2

```
not ( 4 = 4 );
```

false

```
quit();
factor(289);
expand((x+3)^3);
```

`u:(x^3+1)^2;` - assignment
`diff(u,x);` - differentiate
`%e^(%i*%pi);` - constants

3>5;%,pred; - evaluates boolean expression.

`%i` - complex numbers

`cabs(z1)`, `arg(z1)`, `polarform(z1)`, `rectform(z1)`, `conjugate(z1)`

`ev( rectform((5+%i)^0.5), numer);` - force complex number evaluation

`ratprint: false` - suppress rational warnings

`f1(x):=x^2+sin(x); f1(.2);` -defining a function.

```
f04(a,b) := 6*a*b-a+b;
makelist( makelist( f04(a,b),a,1,10 ),b,1,10 );
```

sequence

Maxima: matrix

`#` is not equal to operator.

`evenp(35); FALSE` - even number?

`oddp`

`mod(34,3); 1`

`random(n);` [0,n-1]

`subst(a,b,c);` a for b into c. `subst(5,x,x^2)`

```
integrate(1/x^2,x,1,inf);
(x:1,y:0,for i from 1 thru 10 do (y:y+i),y);
```

`prod(t+i,i,1,5);` - product

```
?round(2.3);

?truncate(ev(5/2,numer));
ev(   ?truncate(   log(15)/log(2)   ),   numer   );
```

`ev(d11,numer);` - numerical evaluation.

`fpprec` - see floating point precision

`fpprec: 40` - set precision

`bfloat(%pi)` - cast number to bfloat

```
   g10(30000, bfloat(7.2));
```

`find_root( w=3/(2+w),w,0,2);` - numerical solver

`append(   [2,3],   [3,5]   );` -> [2,3,3,5]

`w1: append( w1, [[ t0, x0 ]])` - add a point $[t0, x0]$

`is ( 3 = 5 )` -> false, (3==5)

`is ( 3 # 5 )` -> true, (3!=5)

```
e1: x^2+3*x*y+y^2;
e2: 3*x+y=1;
solve([e1,e2]);
solve(3*x+1=sqrt(5),x);
solve([x+y=3,x-y=-1],[x,y])
```

Accessing an equation, use `lhs()` and `rhs()` functions. `r1: solve([e1,e2],[c,d]); rhs( r1[1][1] )`

gave the `c` value.

```
for i:1 while i<=10 do s:s+i;
```

Recursive computation

```
fac[1](x) := 1;
fac[n](x) := n*fac[n-1](x);
```

Alternatively

```
u[0]:5;
u[n] := ev( log(1 + u[n-1]), numer);

f1(n) := if (evenp(n)) then n/2 else 3*n+1;
```

In procedure "," is the statement delimiter, "$" is scope terminator.

```
f02(x) := block
(
  [y],
  y:1,
  while x > 1 do
  (
    y: x*y,
    x: x-1
  ),
  y
)$

f07(n) := block( [s,i], s:0, i:1, while i<n do (s:s+i,
print("s=",s," i=",i),i:i+1), return(s))$

f03() := block( [i], i:1, while i<5 do ( print("i=",i),
i:i-1 ) )$

/* Sophie Germaine test */
f05(n):= block( [],
  if primep(n) then
    if primep(2*n+1) then return(true),
  return(false));

a1[4]:2*y; a1:2
arrayinfo(a1);
```

[HASHED, `length( [1,3,6])` -> 3

```
sort( [3,11,7,2] )
reverse( [3,2,1] )
```

->[1,2,3]

sort with a comparison function. Develop a predicate function, `g3(a,b) := is (g1[a,b] = 0);`, sort([1,2,3,4,5,6,7,8], g3);

`batch("~/file01.txt");` - load file e.g. edited in vi

`save( save01, all );  load(save01)` - session

Random access arrays (effectively)
t3: [3,4,5];
t3[1] - accessing 3
makelist(binomial(6,i),i,0,6);
 [1,6,15,20,15,6,1]

`apply( "+", [3,4,5]);` evaluates 3+4+5
`apply( max, [3,4,5]);  -> 5`
`map(f, [a,b,c] );` evaluates [f(a), f(b), f(c)]
`for x in [a,b,c] do expr`

`floor(4.3)` -> 4
`ceiling(4.3)`-> 5

`taylor( (1+x^(-1))^(1/2),x,0,4)` taylor expansion
`bs(x,n,w) := 1+sum( binomial(n,k)*x^k, k,1,w)`
binomial series expansion
`sum(k^3,k,1,n),simpsum;` - evaluate closed sum

`limit((log(1/x))^x,x,0,plus);` - can include direction

`zeta(2)` - zeta function

Misc

shell (maxima) and gui (wxmaxima) installs

Linux install: `# apt-get install wxMaxima`

Issues

- Command line maxima - entering a procedure there is no way to easily edit the function. **Each command is individually entered and interpreted till the function is complete.** If error occurs then redo - not user friendly. Compare with front end Gui and Mathematica! - document and discuss batch.

Using Maxima as a calculator

wrap expression around in ev(xxx, numer); mostly works.

Example of finding angle and converting answer to degrees. Accessed the previous result variable returned by the system.

```
ev( atan(12./10.), numer);
ev( %o9*180./%pi,numer)
```

`? topic` - help

`primep` - prime test

Links

Harvard Mathematics Department Computing: Maxima

Examples

```
f02(x0,n) := subst(x0,x,taylor(1/(1+x),x,0,n));
subst(0.2,x, taylor(1/(1-x),x,0,3));
```

`limit( (1+x)^(1/2)-x^(1/2),x,inf);` - when this fails substitute a large number for inf

Maxima Manual Plotting

```
plot2d( [x^2, x^3],[x,-5,5]);
plot3d( x^2+y^2,[x,-1,1],[y,-1,1] )
plot3d( x^2+y^2,[x,-1,1],[y,-1,1],[gnuplot_preamble,
"set zero axis"])
```

`plot2d( [discrete, makelist([x,x^2],x,1,3)] );` - lines connecting points

Plotting multiple plots, sampled continuous curve, calculated other set of discrete points separately. Used different point types as one set overwrote the other.

`f02(fn,a,b,n) := makelist(fn(a+x*(b-a)/(n-1)),x,0,n-1);` - sampler

`f07(x) := ev([x, gamma(x+1)],numer);`

`p04: points(f02(f07,2,5,100));` - set of points

`p02:points([ [2,2], [3,6], [4,24], [5,120] ]);` - set of points

```
wxdraw2d( color=red, point_size=2, point_type=4, p02,
color=green, point_size=1,points_joined=true,
point_type=7, p04);
```

wxdraw2d draws inside Maxima as opposed to the external Gnuplot

`tex(%o46);` convert output to latex

`f01(n):=length(divisors(n))` - $\sigma_0(n)$

`setify( [1,2] )` gave `{ 1,2 }` - convert a list to a set

`listify( {1,2} )` gave `[1,2]`

/* sum of divisors */

```
sigma1(n) := apply("+",listify(divisors(n)));
```

/* factor divisors */

```
fdivisors(n) := factor(listify(divisors(n)));
```

`fullratsimp((x^2-2*x+1)/(x-1));` - simplify polynomials

`partfrac(1/(n*n-n),n);` - partial fractions

`minfactorial(n!/(n-2)!);` - simplify factorial expressions

```
f01(n,w,x) := minfactorial(n!/((n-w)!*w!)*x^w);
f02(a,x) := sum(f01(n,k,x),k,0,a);
TODO move
```

TODO - example page, for example combine map and makelist.

```
f01(x) := ev(x/log(x),numer);
map(f01,makelist(1/10^k,k,1,5));
```