Conversation:

WebCAS — A Tutorial

WebCAS is a computer algebra system that runs entirely in your browser. It computes **exactly** where it can (integers of any size, fractions, radicals, symbolic expressions), **numerically** where it must, and shows results as typeset mathematics. This tutorial is a guided tour; every example can be typed directly into the command line.

Examples within a section build on each other. Sections are independent: if a name from an earlier section gets in the way, use **Restart** (or `clear(name)`).

1. Getting started

Action	How
Evaluate	**Enter**
New line	**Shift+Enter**; incomplete input (an open bracket, `if` without `end`) continues automatically
Previous inputs	↑ **/** ↓
Refer to results	`%` (last), `%%`, `%7`; clicking a label such as `%7` inserts it
Abort	**Esc**
Help	**F1**, or type `?integrate`
Results	each has tools: switch between math and text, copy as re-enterable text, copy LaTeX (or pgfplots code for plots), download plots as SVG or PNG
Export	**File** menu: the whole notebook as a script, a LaTeX document, or a session file

The notebook is saved in the browser automatically.

```
1/3 + 1/6                   # 1/2  (exact)
2^100                       # all 31 digits
sqrt(8) + sin(pi/12)        # 2*sqrt(2) + (sqrt(6) - sqrt(2))/4
N(pi, 15)                   # 3.14159265358979
(1 + 2*i)*(3 - i)           # 5 + 5*i
```

Assignments use `:=`, functions are defined like `f(x) := x^2 + 1`, and `=` builds **equations**. To test equality, use `==`. A handy Maxima-style shortcut evaluates and substitutes in one go:

```
diff(x^3*sin(x), x), x = pi        # -pi^3
```

2. Algebra

```
expand((a + b)^5)
factor(x^6 - 1)                    # (x - 1)*(x + 1)*(x^2 + x + 1)*(x^2 - x + 1)
factor(x^4 - y^4)                  # multivariate factorization
together(1/(x - 1) - 1/(x + 1))    # 2/(x^2 - 1)
apart((x^2 + 2)/((x - 1)*(x^2 + 1)), x)
simplify(cos(x)^4 - sin(x)^4)      # cos(2*x)
radsimp(1/(sqrt(2) + sqrt(3)))     # sqrt(3) - sqrt(2)
(x + 1)^3 == x^3 + 3*x^2 + 3*x + 1     # true: identities are proved
```

Number theory comes along for free. Here are the Mersenne primes $2^p - 1$ with $p < 64$:

```
select(range(2, 63), p -> isprime(2^p - 1))      # [2, 3, 5, 7, 13, 17, 19, 31, 61]
ifactor(2^64 + 1)                                # [[274177, 1], [67280421310721, 1]]
```

3. Calculus

```
diff(x^x, x)                                          # (log(x) + 1)*x^x
taylor(tan(x), x, 0, 7)
limit((1 + 1/x)^x, x, inf)                            # e
limit((x - sin(x))/x^3, x, 0)                         # 1/6
integrate(x^2*sin(3*x), x)
integrate(1/(x^3 + 1), x)
integrate(exp(-x^2), x, -inf, inf)                    # sqrt(pi)
integrate(1/(2 + cos(x)), x, 0, 2*pi)                 # 2*sqrt(3)*pi/3
sum(k^3, k, 1, n)                                     # n^2*(n + 1)^2/4
sum(1/(k*(k + 1)), k, 1, inf)                         # 1
sum(k*k!, k, 0, n)                                    # (n + 1)! - 1   (Gosper's
algorithm)
integrate((x^4 - 3*x^2 + 6)/(x^6 - 5*x^4 + 5*x^2 + 4), x) # Sixth-degree denominator
integrate(exp(x)*(1 - x)/(exp(x) - x)^2, x)           # Hermite reduction
integrate(1/(x*log(x)*log(log(x))), x)                # nested logarithms
integrate(log(log(x)), x)                             # an li remainder
integrate((x + 1)^2*exp(x)/x^2, x)                    # an EI remainder
integrate(log(x)^2/(x + 1)^2, x)                      # a partial result
integrate(4^x/(2^x + 1), x)                           # b^x normalization
integrate(exp(x + exp(x)), x)
[iselementary(exp(x^2), x), iselementary(sin(x)/x, x), iselementary(x*exp(x^2), x)]
```

Every antiderivative is checked by differentiation, and every definite integral against numerical integration. WebCAS would rather leave an integral unevaluated than return a wrong answer. When a denominator does not factor,
the result is a sum over its roots (`rootsum`), which you can still evaluate numerically:

```
N(integrate(1/(x^5 - x - 1), x, 2, 3))
```

4. Equations

```
solve(x^3 - 6*x^2 + 11*x - 6 = 0, x)          # [x = 1, x = 2, x = 3]
solve([x^2 + y^2 = 5, x*y = 2], [x, y])       # four solutions
solve(x*exp(x) = 2, x)                         # [x = lambertw(2)]
nsolve(x^5 - x - 1, x)                         # all five roots numerically
nsolve(cos(x) = x, x, 1)                       # Newton from x = 1
subst(first(solve(x^2 = 2, x)), x^4 + x)       # 4 - sqrt(2)
```

5. Linear algebra

Matrices use `.` for the matrix product; `*` is elementwise or scalar.

```
A := matrix([2, 1], [1, 2]);
eigenvects(A)                                  # [[1, 1, [[-1, 1]]], [3, 1, [[1, 1]]]]
factor(det(matrix([1, a, a^2], [1, b, b^2], [1, c, c^2])))   # the Vandermonde product
inverse(matrix([a, b], [c, d]))
nullspace(matrix([1, 2, 3], [4, 5, 6], [7, 8, 9]))
```

Least squares. Fit a line $y = \beta_0 + \beta_1 x$ to the points $(0, 1), (1, 3), (2, 4), (3, 7)$ using the normal equations $\beta = (X^T X)^{-1} X^T y$:

```
X := matrix([1, 0], [1, 1], [1, 2], [1, 3]);
yv := [1, 3, 4, 7];
beta := inverse(transpose(X) . X) . (transpose(X) . yv)      # [9/10, 19/10]
show(listplot([[0, 1], [1, 3], [2, 4], [3, 7]]), plot(beta[1] + beta[2]*x, x, 0, 3))
```

6. Probability and statistics

The standard normal density, its normalization, its variance, and the probability of falling within one standard deviation:

```
pdf(x) := exp(-x^2/2)/sqrt(2*pi);
integrate(pdf(x), x, -inf, inf)          # 1
integrate(x^2*pdf(x), x, -inf, inf)      # 1
N(integrate(pdf(x), x, -1, 1))           # 0.6826894921370857
```

For the exponential distribution, the rate must be positive, which we tell WebCAS with an **assumption**:

```
assume(lambda > 0);
m := integrate(x*lambda*exp(-lambda*x), x, 0, inf)               # 1/lambda
simplify(integrate(x^2*lambda*exp(-lambda*x), x, 0, inf) - m^2)   # 1/lambda^2
forget(lambda);
```

And a classic: the chance of at least one six in four rolls is `1 - (5/6)^4`, i.e. `671/1296`.

7. Plotting

```
plot([sin(x), cos(x)], x, 0, 2*pi, legend = ["sin", "cos"])
plot(tan(x), x, -3, 3)                       # poles are handled
paramplot([cos(3*t), sin(2*t)], t, 0, 2*pi)
contourplot(x^2 - y^2, [x, -2, 2], [y, -2, 2], levels = 12)
densityplot(sin(x)*cos(y), [x, 0, 2*pi], [y, 0, 2*pi])
```

Plots are values: store them, combine them with `show`, and export them as SVG, PNG or pgfplots code.

```
show(plot3d(x^2 - y^2, [x, -1, 1], [y, -1, 1], mesh = false), paramplot3d([cos(t), sin(t),
cos(2*t)], [t, 0, 2*pi]))
```

8. Special functions

WebCAS knows the error, exponential, sine and cosine integrals, incomplete gamma, digamma, Bessel, Airy, Lambert W
and hypergeometric functions, with exact values, derivatives, series and integration rules.

```
integrate(sin(x)/x, x, 0, 1)                        # Si(1)
plot([besselj(0, x), besselj(1, x)], x, 0, 20)
taylor(airyai(x), x, 0, 4)
besselj(1/2, x)                                     # sqrt(2)*sin(x)/(sqrt(pi)*sqrt(x))
```

New functions are easy to define on top of these. The complete elliptic integral of the second kind is a hypergeometric function:

$$E(k) = \frac{\pi}{2}\,{}_2F_1\left(\tfrac{1}{2}, -\tfrac{1}{2}; 1; k^2\right)$$

```
E(x) := pi/2*hyp2f1(1/2, -1/2, 1, x^2);
N(E(1/2))                                  # 1.4674622093394272
```

For $|x| > 1$ the function becomes complex (the branch cut of ${}_2F_1$). Its real part describes the averaged point-spread function of a gravitational lens, as studied for the solar gravitational lens:

```
plot(re(E(x)), x, -3, 3)
```

9. Programming

The language has `if/elif/else`, `for`, `while`, `block` with `local` variables, lists, and recursion:

```
fib(n) := if n <= 1 then n else fib(n - 1) + fib(n - 2) end;
table(fib(k), k, 0, 15)

collatz(n) := block(local c := 0;
                    while n > 1 do
                      n := if mod(n, 2) == 0 then n/2 else 3*n + 1
                    end;
                    c := c + 1 end; c);
map(collatz, range(1, 10))    # [0, 1, 7, 2, 5, 8, 16, 3, 19, 6]
```

User functions, including recursive ones and blocks with local tables, can be plotted directly. Here is a complex-valued gamma function via the Lanczos approximation, with the reflection formula for the left half-plane. Its reciprocal shows the poles at $0, -1, -2, \ldots$ as dark spots:

```
Gam(z) := if re(z) < 0.5 then pi/sin(pi*z)/Gam(1 - z)
          else
            block(local LANCZOS := [0.99999999999980993, 676.5203681218851,
-1259.1392167224028,
                                    771.32342877765313, -176.61502916214059,
12.507343278686905,
                                    -0.13857109526572012, 9.9843695780195716e-6,
1.5056327351493116e-7],
                        a := LANCZOS[1], t := z + 6.5, h := t^((z - 0.5)/2);
                  for k from 2 to 9 do a := a + LANCZOS[k]/(z + k - 2) end;
                  sqrt(2*pi)*h*exp(-t)*h*a)
          end;
Gam(5.0)                                          # 24.000000000000064
densityplot(1/abs(Gam(x + i*y/5)), [x, -2, 2], [y, -2, 2])
```

10. Differential equations

Symbolic ODEs use prime notation, with conditions in the same list:

```
dsolve([y''(x) + 2*y'(x) + 5*y(x) = 0, y(0) = 1, y'(0) = -1], y(x))    # cos(2*x)*exp(-x)
dsolve(y'(x) = x*y(x)^2, y(x))
dsolve(x^2*y''(x) + x*y'(x) + (x^2 - n^2)*y(x) = 0, y(x))           # Bessel functions
dsolve([y''(x) + y(x) = heaviside(x - 1), y(0) = 0, y'(0) = 0], y(x))   # via Laplace
transforms
dsolve([u'(t) = v(t), v'(t) = -u(t), u(0) = 1, v(0) = 0], [u(t), v(t)])
```

Numerically, for example the nonlinear pendulum:

```
P := rhs(first(ndsolve([y''(x) + sin(y(x)) = 0, y(0) = 1, y'(0) = 0], y(x), [x, 0, 10])));
plot(P, x, 0, 10)
```

Partial differential equations: characteristics, Fourier series, and the method of lines:

```
pdsolve(diff(u(x, t), t) + u(x, t)*diff(u(x, t), x) = 0, u(x, t))      # Burgers: u = F(x -
t*u)
heat := [diff(u(x, t), t) = diff(u(x, t), x, 2), u(0, t) = 0, u(1, t) = 0, u(x, 0) = x*(1 -
x)];
pdsolve(heat, u(x, t))                          # a symbolic Fourier series
```

```
U := rhs(first(npdsolve(heat, u(x, t), [x, 0, 1], [t, 0, 0.1])));
densityplot(U, [x, 0, 1], [t, 0, 0.1], labels = ["x", "t"])
```

11. General relativity: the Schwarzschild solution

WebCAS is powerful enough for component calculations in general relativity. We define the Christoffel symbols of the first kind (`lcs`) and second kind (`mcs`), and the Riemann tensor (`R`), for any metric `g` in coordinates `c`:

```
lcs(g,c):=block(local d:=length(c),r:=table(zeros(d,d),k,1,d);
  for k from 1 to d do
    for l from 1 to d do
      for m from 1 to d do
        r[k][l][m]:=1/2*(diff(g[k][l],c[m])+diff(g[k][m],c[l])-diff(g[l][m],c[k]))
      end
    end
  end;
r);
mcs(g,c):=block(local G:=g^-1,L:=lcs(g,c); table(sum(G[k,l]*L[k],k,1,4),l,1,4));
R(g,c):=block(
  local d:=length(c),
        G:=mcs(g,c),
        R:=table(table(zeros(d,d),k,1,d),l,1,d);
  for k from 1 to d do
    for l from 1 to d do
      for m from 1 to d do
        for n from 1 to d do
          R[k][n][m][l]:=diff(G[k][l][m],c[n])-
                         diff(G[k][l][n],c[m])-
                         sum(G[a][l][n]*G[k][a][m],a,1,d)+
                         sum(G[a][l][m]*G[k][a][n],a,1,d)
        end
      end
    end
  end;
R);
```

Now take a general static, spherically symmetric metric with two unknown functions $A(r)$ and $B(r)$, declared with `depends`:

```
clear(A);
depends([A,B],r);
lg:=diagmatrix([B,-A,-r^2,-r^2*sin(theta)^2]);
coords:=[t,r,theta,phi];
```

Contracting the Riemann tensor gives the Ricci tensor; with one index raised and the trace removed, we obtain the Einstein tensor $G^{\mu}{}_{\nu}$ (named `Ein` here, since `E` was used for the elliptic integral above):

```
S:=simplify(sum(R(lg,coords)[k][k],k,1,4).lg^-1);
Ein:=simplify(S-1/2*trace(S)*identity(4))
```

The vacuum equation $G^{t}{}_{t} = 0$ is a first-order ODE for A. Solving it, and naming the integration constant so that it matches the Newtonian limit, recovers $g_{rr} = -(1 - 2GM/r)^{-1}$:

```
T:=solve(exp(logcombine(subst(C1=log(2*G*M),dsolve(numer(Ein[1,1])=0,A)),"positive")),A)
```

Finally, a picture of A in the equatorial plane. With $G = 1/(10M)$ the horizon sits at $r = 1/5$:

```
densityplot(subst([G=1/(10*M),r=sqrt(x^2+y^2)],rhs(T[1])),[x,-10,10],[y,-10,10])
```

This takes a few seconds: it differentiates and simplifies several hundred tensor components symbolically.

12. Assumptions, dependencies and declared derivatives

```
assume(a > 0);
sqrt(a^2)                                   # a
integrate(exp(-a*x^2), x, -inf, inf)        # sqrt(pi)/sqrt(a)
dsolve(y''(x) + a^2*y(x) = 0, y(x))         # C1*cos(a*x) + C2*sin(a*x)
forget(a);

depends(f, x); diff(f^2, x)                 # 2*diff(f, x)*f
gradef(g(x), h(x)); integrate(2*x*h(x^2), x)    # g(x^2)
```

13. Where to go from here

Press **F1** for the full reference: every function has a description and a runnable example. Use **File → Download LaTeX** to turn a notebook into a paper-ready document, with plots as pgfplots figures. And if something isn't evaluated, WebCAS is telling you honestly that it doesn't know. That is always better than a wrong answer.