Performance Updates in Maple 2026

Operations with Units

- Performance of calculations involving quantities with Units has been greatly improved in Maple 2026. Big improvements have been made with the performance of comparisons piecewise and core operations.
- For example, the following example now takes 0.028s in Maple 2026 compared to 2.532s in Maple 2025.0. This represents over a 90 times speedup.

```
> with(Units:-Simple):
```

```
> p := proc(a)
     if a > 0 and a < 1*Unit(m) or a = undefined then
        0;
     elif a > 10*Unit(m) and a < 50*Unit(m) then
        1;
     else
        -1;
     end if;
  end proc:
```

```
> tt := time():
  for i to 10^3 do
      p(0.5*Unit(m))+p(5*Unit(m))+p(100*Unit(m));
  end do:
  time()-tt;
```

- The following example is a naive implementation that calculates the nth Fibonacci number. Because there is no remember table or caching in general, the running time is exponential in **n**. This benchmark with n=35 results in 242785 function calls, and is a measure of general overhead. Maple 2026 completes this loop in 0.192s, whereas with Units:-Simple loaded in Maple 2025.0 it took 0.462s.

```
> with(Units:-Simple):
```

```
> N := 25:
```

```
> F := proc(n)
    if n < 2 then n
    else F(n-1) + F(n-2);
    fi;
```

```
  end proc:
> tt := time();
> F(N);
> time() - tt;
```

Hardware-Float Vector Initialization

- Creation of double-precision float[8] vectors has been optimized when using an initializer of the form p->c1*F(ca2*p+c3)+c4, where F is a known unary evalhf function and ci's are numeric constants. The following example is 50% faster in Maple 2026 compared to Maple 2025.

```
> restart;
> N:=10^6:
> tt:=time[real]():
> signal:=Vector(N,i->2*sin(i+1),datatype=float[8]):
> time[real]()-tt;
```

- Initializers in this form that call evalhf directly have also been optimized to avoid the overhead of entering the evalhf subsystem at each initialization point. The following example has identical timings to the above example, and is over 10 times faster than Maple 2025.

```
> restart;
> N:=10^6:
> tt:=time[real]():
> signal:=Vector(N,i->evalhf(2*sin(i+1)),datatype=float[8]):
> time[real]()-tt;
```

- Also improved is the case of p->evalhf(F(c2*p+c3)), and various forms of F, *, and + when Units:-Simple is loaded. The following example is over 10 times faster in Maple 2026.

```
> restart;
> with(Units:-Simple):
> N:=10^6:
```

```
> tt:=time[real]():
> signal:=Vector(N,i->evalhf(sin(i+1)),datatype=float[8]):
> time[real]()-tt;
```

XMLTools:-XMLElement

- The XMLTools package is an extensive package for dealing with XML content. The XMLElement command constructs an XML tag in the representation used by the package. For Maple 2026, we sped this command up significantly. The example below executes about 3x as quickly as before.

```
> with(XMLTools):
> children := map2(XMLElement, "abc", [["foo" = "bar"], [], ["baz" =
  "bar"]], ["str0"]);
> CodeTools:-Usage(XMLElement("abc", ["def" = "ghi", "hij" = "klm"],
  ["nop", children[1], "qrs", children[2], "tuv", children[3]]),
  iterations=10^4);
```

Polynomial system solving and Rational Univariate Representation

- The Groebner[RationalUnivariateRepresentation] command has a new sub-algorithm which uses parallel computation and is considerably faster for large systems. It can be invoked with the new option **'method'='ZDS'**; the previous implementation is still available via **'method'='RS'**.
- The new implementation is automatically used under the hood by the RootFinding[Isolate] command for solving polynomial systems.
- As an example, we compare the running time of the **RationalUnivariateRepresentation** command for the Katsura-7 example with both methods. The benchmark was run on an Apple M1 Pro processor with 10 cores.

```
> katsura7 := [x0^2+2*x1^2+2*x2^2+2*x3^2+2*x4^2+2*x5^2+2*x6^2+2*
  x7^2+2*x[8]^2-x0,
              2*x0*x1+2*x1*x2+2*x2*x3+2*x3*x4+2*x4*x5+2*x5*x6+2*x6*
  x7+2*x7*x[8]-x1,
              2*x0*x2+x1^2+2*x1*x3+2*x2*x4+2*x3*x5+2*x4*x6+2*x5*
  x7+2*x6*x[8]-x2,
              2*x0*x3+2*x1*x2+2*x1*x4+2*x2*x5+2*x3*x6+2*x4*x7+2*x5*
  x[8]-x3,
              2*x0*x4+2*x1*x3+2*x1*x5+x2^2+2*x2*x6+2*x3*x7+2*x4*x
  [8]-x4,
              2*x0*x5+2*x1*x4+2*x1*x6+2*x2*x3+2*x2*x7+2*x3*x[8]-x5,
              2*x0*x6+2*x1*x5+2*x1*x7+2*x2*x4+2*x2*x[8]+x3^2-x6,
```

```
                2*x0*x7+2*x1*x6+2*x1*x[8]+2*x2*x5+2*x3*x4-x7,
                2*x[8]+2*x7+2*x6+2*x5+2*x4+2*x3+2*x2+2*x1+x0-1]:

> CodeTools:-Usage(Groebner:-RationalUnivariateRepresentation
  (katsura7,'method'='RS'), 'output'='realtime', 'quiet')*s;

> CodeTools:-Usage(Groebner:-RationalUnivariateRepresentation
  (katsura7,'method'='ZDS'), 'output'='realtime', 'quiet')*s;
```

- Below is a table of timings for various benchmark problems, both without and with option `'output'='factored'`. If a table cell is empty, then the corresponding computation was not attempted because it would take longer than 1 minute. For the first 11 benchmark problems, `'output'='factored'` was not done because the RUR does not factor for these examples. The **Dimension** column gives the total number of complex solutions, which equals the degree of the univariate polynomial in the RUR.

Apple M1 Pro 10 cores	Dimension	Time RS (s)	Time ZDS (s)	Speedup	Factor degrees	Time RS factored (s)	Time ZDS factored (s)	Speedup
Eco(9)	128	0.61	1.08	0.56	128	----	----	----
Eco(10)	256	4.67	1.60	2.91	256	----	----	----
Eco(11)	512	56.18	7.68	7.32	512	----	----	----
Eco(12)	1024	----	64.88	----	1024	----	----	----
Chandra(7)	64	0.17	0.45	0.37	64	----	----	----
Chandra(8)	128	1.52	2.53	0.60	128	----	----	----
Chandra(9)	256	18.15	7.74	2.35	256	----	----	----
Chandra (10)	512	----	52.35	----	512	----	----	----
Reimer(4)	36	0.02	0.01	1.62	36	----	----	----
Reimer(5)	144	1.20	0.25	4.80	144	----	----	----
Reimer(6)	576	----	3.78	----	576	----	----	----
Katsura(6)	128	1.51	0.50	3.04	1•2+126•1	1.64	1.00	1.65

Katsura(7)	256	18.16	1.97	9.23	1•2+2•1+12•1+240•1	18.67	2.30	8.12
Katsura(8)	512	----	8.63	----	1•2+6•1+504•1	----	12.60	----
Katsura(9)	1024	----	69.56	----	1•2+2•1+30•1+990•1	----	90.66	----
Cyclic(5)	70	0.03	0.03	1.26	2•5+4•15	0.05	0.10	0.52
Cyclic(6)	156	0.78	0.50	1.54	2•12+4•15+8•9	0.89	0.25	3.56
Cyclic(7)	924	----	8.93	----	2•21+6•35+12•14+24•21	----	7.87	----
Root(5)	120	0.16	0.19	0.84	4•30	0.19	0.18	1.05
Root(6)	720	----	3.66	----	4•180	----	4.60	----
Noon(4)	73	0.10	0.06	1.77	3•1+4•4+6•9	0.14	0.11	1.35
Noon(5)	233	6.43	1.02	6.28	3•1+4•5+6•35	7.15	1.46	4.89
Noon(6)	717	----	14.87	----	3•1+4•6+6•115	----	13.05	----
Schwarz(7)	128	0.99	0.45	2.19	1•4+2•2+3•4+6•18	1.11	0.46	2.39
Schwarz(8)	256	13.64	1.76	7.77	1•16+2•40+4•40	15.49	2.06	7.52
Schwarz(9)	512	----	4.95	----	1•8+2•12+3•8+6•76	----	10.73	----
Schwarz (10)	1024	----	30.64	----	1•16+2•24+4•240	----	27.13	----
Virasoro	256	21.91	7.62	2.88	1•144+2•40+4•8	25.73	7.19	3.58