Performance Improvements in Maple 2024

Operations on Matrices Containing Quantities with Units

- A new Units indexing function has been created which allows a matrix, vector, or array to keep units separate from the data. This allows the data to be a single block of hardware type values, including datatype=float[8] and datatype=complex[8]. LinearAlgebra and other package routines can then dispatch to fast hardware algorithms providing optimal performance.
- Here is an example that creates a 100x100 matrix of random data, each element has the unit of meters per second. Computing the inverse takes milliseconds, which is over a thousand times faster than the way done in previous versions which required datatype=anything.

```
> M := LinearAlgebra:-RandomMatrix(100,datatype=float[8],shape=Unit
  (m/s)):
```

```
> CodeTools:-Usage( LinearAlgebra:-MatrixInverse(M) ):
```

```
memory used=413.27KiB, alloc change=0 bytes, cpu time=33.00ms, real
time=178.00ms, gc time=0ns
```

- A matrix with the Units indexing function can be used seamlessly in place of a matrix that has no indexing function, but keeps quantities with units. Such a matrix allows a single unit for all data, different units on different entries, and even entries with no unit assigned. The matrix will adapt to the situation.

```
> M := Matrix( 2, (i,j)->(10*i+j)*Unit(kg), shape=Unit(kg) );
```

$$M := \begin{bmatrix} 11\ \mathrm{kg} & 12\ \mathrm{kg} \\ 21\ \mathrm{kg} & 22\ \mathrm{kg} \end{bmatrix}$$

```
> M[1,1] := 1.23 * Unit(ohm);
```

$$M_{1,1} := 1.23\ \Omega$$

```
> M[1,2] := 17;
```

$$M_{1,2} := 17$$

```
> M;
```

$$\begin{bmatrix} 1.23\ \Omega & 17 \\ 21\ \mathrm{kg} & 22\ \mathrm{kg} \end{bmatrix}$$

- Quantities assigned into the matrix will be converted into the default unit specified for that matrix whenever possible.

```
> M[2,2] := 35.2 * Unit(lb);
```

$$M_{2,2} := 35.2\ \mathrm{lb}$$

```
> M;
```

$$\begin{bmatrix} 1.23\,\Omega & 17 \\ 21\,\mathrm{kg} & 15.96645142\,\mathrm{kg} \end{bmatrix}$$

- The unit and data can be easily separated, so algorithms can inspect and take advantage of the new structure as needed. In this example we perform a matrix power by operating on just the data, and only apply a unit transformation at the end.   This kind of manipulation is used behind the scenes in many built-in operations in the LinearAlgebra package.

```
> A := Matrix(2,(i,j)->(10*i+j)*Unit(m), shape=Unit(m), datatype=float
  [8] );
```

$$A := \begin{bmatrix} 11.\,\mathrm{m} & 12.\,\mathrm{m} \\ 21.\,\mathrm{m} & 22.\,\mathrm{m} \end{bmatrix}$$

```
> default_unit, data := op(rtable_split_unit(A));
```

$$\mathit{default\_unit}, \mathit{data} := \mathrm{m}, \begin{bmatrix} 11. & 12. \\ \vdots & \vdots \end{bmatrix}$$

```
> newA := rtable_set_indfn(data^4,Units:-Simple:-simplify
  (default_unit^4));
```

$$\mathit{newA} := \begin{bmatrix} 413557.\,\mathrm{m}^4 & 439164.\,\mathrm{m}^4 \\ 768537.\,\mathrm{m}^4 & 816124.\,\mathrm{m}^4 \end{bmatrix}$$

- The Units:-Simple package makes use of this new data structure by default when possible.

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

[`*`, `+`, `-`, `/`, `<`, <=, <>, `=`, `^`, arccos, arccosh, arccot, arccoth, arccsc, arccsch, arcsec, arcsech, arcsin, arcsinh, arctan, arctanh, argument, *collect*, *combine*, cos, cosh, cot, coth, csc, csch, *diff*, *eval*, *evalc*, *evalr*, exp, *factor*, *frem*, ln, log, log10, *log2*, max, min, *piecewise*, polar, *root*, sec, sech, *shake*, sin, sinh, sqrt, surd, tan, tanh, *type*, *verify*]

```
> A := <1,2;3,4>*Unit(m);
```

$$A := \begin{bmatrix} \mathrm{m} & 2\,\mathrm{m} \\ 3\,\mathrm{m} & 4\,\mathrm{m} \end{bmatrix}$$

```
> B := <5,6;7,8>*Unit(s);
```

$$B := \begin{bmatrix} 5\,\mathrm{s} & 6\,\mathrm{s} \\ 7\,\mathrm{s} & 8\,\mathrm{s} \end{bmatrix}$$

```
> C := A . B^(-1);
```

$$C := \begin{bmatrix} 3\,\frac{m}{s} & -2\,\frac{m}{s} \\ 2\,\frac{m}{s} & -\frac{m}{s} \end{bmatrix}$$

```
> rtable_indfns(A);
```

$$m$$

```
> rtable_indfns(B);
```

$$s$$

```
> rtable_indfns(C);
```

$$\frac{m}{s}$$

- For additional improvements involving units in Maple 2024, see Improvements in Handling of Units in Maple 2024.

Floating-Point Matrix and Vector Initialization

- Maple uses hardware double-precision floating-point representations when constructing matrices and vectors specified with datatype=float[8]. On initialization Maple 2024 calls directly to the evalhf implementation of known functions and avoids conversion from hardware to software floats. Performance of examples such as the following has dramatically sped up.

```
> tt:=time[real]():
> signal:=Vector(10^5,i->sin(i*1.0),datatype=float[8]):
> time[real]()-tt;
```

$$0.425$$

Faster Univariate Complex Solver

- Maple 2024 contains a new solver for isolating and approximating all complex roots of a univariate polynomial with *numeric* or *complex(numeric)* coefficients. This solver is available through the method=PW option of the RootFinding:-Isolate command. Except when the domain option is given, the new solver is considerably faster than the one used when method=HR is specified, in particular for high accuracy, and it is now the default when option complex is used. Here are a few examples and timing comparisons for the two solvers.
- Random polynomial, increasing accuracy.

```
> f := randpoly(x, dense, coeffs=rand(-2^32+1..2^32-1), degree=512):
```

```
> CodeTools:-Usage(RootFinding:-Isolate(f, digits=10, method=HR)):
  CodeTools:-Usage(RootFinding:-Isolate(f, digits=10, method=PW)):
memory used=6.80MiB, alloc change=0 bytes, cpu time=7.29s, real time=
7.38s, gc time=0ns
memory used=5.61MiB, alloc change=0 bytes, cpu time=1.89s, real time=
1.94s, gc time=0ns
> CodeTools:-Usage(RootFinding:-Isolate(f, digits=15, method=HR)):
  CodeTools:-Usage(RootFinding:-Isolate(f, digits=15, method=PW)):
memory used=5.20MiB, alloc change=0 bytes, cpu time=8.18s, real time=
8.18s, gc time=0ns
memory used=5.61MiB, alloc change=0 bytes, cpu time=1.96s, real time=
1.96s, gc time=0ns
> CodeTools:-Usage(RootFinding:-Isolate(f, digits=30, method=HR)):
  CodeTools:-Usage(RootFinding:-Isolate(f, digits=30, method=PW)):
memory used=5.42MiB, alloc change=0 bytes, cpu time=8.32s, real time=
8.32s, gc time=0ns
memory used=5.57MiB, alloc change=0 bytes, cpu time=1.47s, real time=
1.47s, gc time=0ns
```

- Bernoulli polynomials, increasing degree.

```
> g1 := expand(bernoulli(128,x)):
> CodeTools:-Usage(RootFinding:-Isolate(g1, digits=10, method=HR)):
  CodeTools:-Usage(RootFinding:-Isolate(g1, digits=10, method=PW)):
memory used=1.57MiB, alloc change=0 bytes, cpu time=13.20s, real time=
13.20s, gc time=0ns
memory used=1.35MiB, alloc change=0 bytes, cpu time=2.00s, real time=
2.00s, gc time=0ns
> g2 := expand(bernoulli(255,x)):
> CodeTools:-Usage(RootFinding:-Isolate(g2, digits=10, method=HR)):
  CodeTools:-Usage(RootFinding:-Isolate(g2, digits=10, method=PW)):
memory used=3.61MiB, alloc change=0 bytes, cpu time=65.20s, real time=
65.21s, gc time=0ns
memory used=2.94MiB, alloc change=0 bytes, cpu time=4.63s, real time=
4.63s, gc time=0ns
```

RootFinding

- The EvaluateAtRoot command in the RootFinding package has a new option \`avoidsymbolic\`. Its main purpose is to improve the command's efficiency. When specified, symbolic processing to ascertain whether the given root is an exact zero of a constraint or not will be skipped. In general this will lead to weaker answers, e.g., *FAIL* being returned instead of *true* or *false*.

- In the following example, the unique root of *sys* contained in *sysroot* is actually an exact root of the non-strict inequality in *cons*.

```
> vars := [x, y];
```

$$vars := [x, y]$$

```
> sys := [-7*x^5+22*x^4-55*x^3-94*x^2+87*x-56,
          -10*y^5+62*y^4-82*y^3+80*y^2-44*y+71];
```

$$sys := [-7x^5 + 22x^4 - 55x^3 - 94x^2 + 87x - 56, -10y^5 + 62y^4 - 82y^3 + 80y^2 - 44y + 71]$$

```
> sysroot := [[-228940483906911495/144115188075855872,
  -57235120976727867/36028797018963968],
               [338902091289333/70368744177664,
  677804182578693/140737488355328]];
```

$$sysroot := \left[\left[-\frac{228940483906911495}{144115188075855872}, -\frac{57235120976727867}{36028797018963968}\right], \left[\frac{338902091289333}{70368744177664}, \frac{677804182578693}{140737488355328}\right]\right]$$

```
> cons := [434*x^9-2043*x^8+6055*x^7-1085*x^6-10004*x^5+17167*x^4
  -6842*x^3+11542*x^2-6997*x <= -4648,
           -50*y^5+23*y^4+75*y^3-92*y^2+6*y+74 > 0];
```

$$cons := [434x^9 - 2043x^8 + 6055x^7 - 1085x^6 - 10004x^5 + 17167x^4 - 6842x^3 + 11542x^2 - 6997x \le -4648, 0 < -50y^5 + 23y^4 + 75y^3 - 92y^2 + 6y + 74]$$

```
> consSet := convert(cons, 'set');
```

$$consSet := \{434x^9 - 2043x^8 + 6055x^7 - 1085x^6 - 10004x^5 + 17167x^4 - 6842x^3 + 11542x^2 - 6997x \le -4648, 0 < -50y^5 + 23y^4 + 75y^3 - 92y^2 + 6y + 74\}$$

- By default, EvaluateAtRoot uses symbolic computation to prove that the first constraint is satisfied:

```
> CodeTools:-Usage(RootFinding:-EvaluateAtRoot(cons, sysroot, sys,
  vars));
memory used=3.14MiB, alloc change=0 bytes, cpu time=62.00ms, real time=
75.00ms, gc time=0ns
```

$$[true, false]$$

```
> CodeTools:-Usage(RootFinding:-EvaluateAtRoot(consSet, sysroot, sys,
  vars));
memory used=446.36KiB, alloc change=0 bytes, cpu time=10.00ms, real
time=10.00ms, gc time=0ns
```

$$false$$

- When specifying the avoidsymbolic option and giving a maximum value of 100 for the floating-point precision, the underlying numerical computation will not be able to prove that the first constraint is satisfied, and therefore *FAIL* is returned:

```
> CodeTools:-Usage(RootFinding:-EvaluateAtRoot(cons, sysroot, sys,
  vars, 'avoidsymbolic' = true, 'threshold' = 100));
memory used=143.32KiB, alloc change=0 bytes, cpu time=3.00ms, real time=
3.00ms, gc time=0ns
```

$$[FAIL, false]$$

- Nevertheless, in the following calling sequence, where we are only interested in the truth value of the logical conjunction of the two constraints, this weaker result is still sufficient, since the 2nd constraint, and therefore the whole conjunction, is *false*:

```
> CodeTools:-Usage(RootFinding:-EvaluateAtRoot(consSet, sysroot, sys,
  vars, 'avoidsymbolic' = true, 'threshold' = 100));
memory used=133.91KiB, alloc change=0 bytes, cpu time=3.00ms, real time=
3.00ms, gc time=0ns
```

$$false$$

Quantifier Elimination

- Significant performance improvements have been made to the QuantifierElimination package, including faster evaluation at sample points by using RootFinding:-EvaluateAtRoot and caching of intermediate results.
- The following examples are 6-7 times faster than Maple 2023.

```
> formula1 := forall([x,y,t], Implies(And(x^3+y^2-x=t, t^2=4/27, t<0),
  x^2+y^2>=rho));
```

$$formula1 := \forall\left([x, y, t], x^3 + y^2 - x = t \wedge t^2 = \frac{4}{27} \wedge t < 0 \quad \rho \leq x^2 + y^2\right)$$

```
> CodeTools:-Usage(QuantifierElimination:-QuantifierEliminate
  (formula1));
memory used=2.89GiB, alloc change=191.09MiB, cpu time=76.62s, real time=
65.76s, gc time=13.37s
```

$$\rho - RootOf\left(729\_Z^2 + 270\_Z - 83, -\frac{673038004593441041303}{1180591620717411303424} .. -\frac{2692152018373764165185}{4722366482869645213696}\right) < 0 \vee \rho - RootOf\left(729\_Z^2 + 270\_Z - 83, -\frac{673038004593441041303}{1180591620717411303424} .. -\frac{2692152018373764165185}{4722366482869645213696}\right) = 0 \vee \left(RootOf\left(729\_Z^2 + 270\_Z - 83, -\frac{673038004593441041303}{1180591620717411303424} .. -\frac{2692152018373764165185}{4722366482869645213696}\right) - \rho < 0 \wedge \rho - RootOf\left(27\_Z^2\right.\right.$$

$$- 18\,\_Z - 13, -\frac{64411164140810191969}{147573952589676412928}\,..-\frac{1030578626252963071477}{2361183241434822606848}\Big) < 0\Big) \vee \rho$$

$$- \mathit{RootOf}\Big(27\,\_Z^2 - 18\,\_Z - 13, -\frac{64411164140810191969}{147573952589676412928}\,..-\frac{1030578626252963071477}{2361183241434822606848}\Big)$$

$$= 0 \vee \Big(\mathit{RootOf}\Big(27\,\_Z^2 - 18\,\_Z - 13, -\frac{64411164140810191969}{147573952589676412928}\,..$$

$$-\frac{1030578626252963071477}{2361183241434822606848}\Big) - \rho < 0 \wedge \rho - \mathit{RootOf}\Big(27\,\_Z^2 - 4,$$

$$-\frac{1817639706731237273789}{4722366482869645213696}\,..-\frac{908819853365618636881}{2361183241434822606848}\Big) < 0\Big) \vee \rho - \mathit{RootOf}\Big(27\,\_Z^2 - 4,$$

$$-\frac{1817639706731237273789}{4722366482869645213696}\,..-\frac{908819853365618636881}{2361183241434822606848}\Big) = 0 \vee \Big(\mathit{RootOf}\Big(27\,\_Z^2 - 4,$$

$$-\frac{1817639706731237273789}{4722366482869645213696}\,..-\frac{908819853365618636881}{2361183241434822606848}\Big) - \rho < 0 \wedge 27\,\rho < -5\Big) \vee 27\,\rho$$

$$+ 5 = 0 \vee \Big(-27\,\rho < 5 \wedge \rho - \mathit{RootOf}\Big(27\,\_Z^2 - 18\,\_Z - 1, -\frac{30439693221836108653}{590295810358705651712}\,..$$

$$-\frac{1217587728873444434585}{2361183241434822606848}\Big) < 0\Big) \vee \rho - \mathit{RootOf}\Big(27\,\_Z^2 - 18\,\_Z - 1,$$

$$-\frac{30439693221836108653}{590295810358705651712}\,..-\frac{1217587728873444434585}{2361183241434822606848}\Big) = 0 \vee \Big(\mathit{RootOf}\Big(27\,\_Z^2 - 18\,\_Z$$

$$- 1, -\frac{30439693221836108653}{590295810358705651712}\,..-\frac{1217587728873444434585}{2361183241434822606848}\Big) - \rho < 0 \wedge \rho < 0\Big) \vee \rho = 0$$

$$\vee \Big(-\rho < 0 \wedge \rho - \mathit{RootOf}\Big(729\,\_Z^2 + 270\,\_Z - 83, \frac{1886254790177420764705}{9444732965739290427392}$$

$$..\frac{471563697544355191183}{2361183241434822606848}\Big) < 0\Big) \vee \rho - \mathit{RootOf}\Big(729\,\_Z^2 + 270\,\_Z - 83,$$

$$\frac{1886254790177420764705}{9444732965739290427392}\,..\frac{471563697544355191183}{2361183241434822606848}\Big) = 0 \vee \Big(\mathit{RootOf}\Big(729\,\_Z^2 + 270\,\_Z$$

$$- 83, \frac{1886254790177420764705}{9444732965739290427392}\,..\frac{471563697544355191183}{2361183241434822606848}\Big) - \rho < 0 \wedge 3\,\rho < 1\Big) \vee 3\,\rho - 1$$

$$= 0$$

```
> formula2 := exists([t__10,t__11,t__12,t__6,b__12,t__5], And(
     0<=t__10, 0<=t__11, 0<=t__12, 0<=t__6, 0<=b__12,
     t__10<=1, t__11<=1, t__12<=1, t__6<=1, b__12<=1,
     0<t__5, t__5<1, t__5^3-2*t__5^2+t__5<0, 0<t__5^3-2*t__5^2+
  t__5+1,
     t__6+1-t__5=0, -t__5^2+t__10+t__5=0, t__5^2-t__11+t__12-2*
  t__5+1=0,
     -t__5^2+t__11+2*t__5+b__12-2 = 0));
```

$$formula2 := \exists\left(\left[t_{10}, t_{11}, t_{12}, t_6, b_{12}, t_5\right], 0 \le t_{10} \wedge 0 \le t_{11} \wedge 0 \le t_{12} \wedge 0 \le t_6 \wedge 0 \le b_{12} \wedge t_{10} \le 1 \wedge t_{11} \le 1 \wedge t_{12} \le 1 \wedge t_6 \le 1 \wedge b_{12} \le 1 \wedge 0 < t_5 \wedge t_5 < 1 \wedge t_5^3 - 2t_5^2 + t_5 < 0 \wedge 0 < t_5^3 - 2t_5^2 + t_5 + 1 \wedge t_6 + 1 - t_5 = 0 \wedge -t_5^2 + t_{10} + t_5 = 0 \wedge t_5^2 - t_{11} + t_{12} - 2t_5 + 1 = 0 \wedge -t_5^2 + b_{12} + t_{11} + 2t_5 - 2 = 0\right)$$

```
> CodeTools:-Usage(QuantifierElimination:-QuantifierEliminate
  (formula2));
```

```
memory used=2.59GiB, alloc change=0 bytes, cpu time=72.46s, real time=
62.72s, gc time=11.90s
```

$$false$$

- In addition, the display of various data structures in the QuantifierElimination package has been improved.

```
> formula := exists(x, x^2+b*x+c=0);
```

$$formula := \exists\left(x, b\,x + x^2 + c = 0\right)$$

```
> C := QuantifierElimination:-PartialCylindricalAlgebraicDecompose
  (formula, 'output'=['data']);
```

$$C := \left[\begin{array}{lcl} \mathit{Variables} & = & [c, b, x] \\ \mathit{Input\ Formula} & = & \exists\left(x, b\,x + x^2 + c = 0\right) \\ \#\,\mathit{Cells} & = & 17 \\ \mathit{Projection\ polynomials\ for\ level\ 1} & = & \left[\ \vdots\ \right] \\ \mathit{Projection\ polynomials\ for\ level\ 2} & = & \left[\ b^2 - 4\,c\ \right] \\ \mathit{Projection\ polynomials\ for\ level\ 3} & = & \left[\ b\,x + x^2 + c\ \right] \end{array}\right.$$

```
> GetLeafCells(C)[1..5];
```

$$\left[\begin{bmatrix} \mathit{Description} & = & c<0 \land x<\mathit{RootOf}\left(\_Z^2+b\_Z+c, \mathit{index}=\mathit{real}_1\right) \\ \mathit{Sample\ Point} & = & [c=-1, b=0, x=-2] \\ \mathit{Index} & = & [1,1,1] \end{bmatrix}, \begin{bmatrix} \mathit{Description} & = & c<0 \land x=\mathit{RootOf}\left(\_Z^2+b\_Z+c, \mathit{index}=\mathit{real}_1\right) \\ \mathit{Sample\ Point} & = & [c=-1, b=0, x=-1] \\ \mathit{Index} & = & [1,1,2] \end{bmatrix}, \begin{bmatrix} \mathit{Description} & = & c=0 \land b<0 \land x<\mathit{RootOf}\left(\_Z\left(\_Z+b\right), \mathit{index}=\mathit{real}_1\right) \\ \mathit{Sample\ Point} & = & [c=0, b=-1, x=-1] \\ \mathit{Index} & = & [2,1,1] \end{bmatrix}, \begin{bmatrix} \mathit{Description} & = & c=0 \land b<0 \land x=\mathit{RootOf}\left(\_Z\left(\_Z+b\right), \mathit{index}=\mathit{real}_1\right) \\ \mathit{Sample\ Point} & = & [c=0, b=-1, x=0] \\ \mathit{Index} & = & [2,1,2] \end{bmatrix}, \begin{bmatrix} \mathit{Description} & = & c=0 \land b=0 \land x<0 \\ \mathit{Sample\ Point} & = & [c=0, b=0, x=-1] \\ \mathit{Index} & = & [2,2,1] \end{bmatrix}\right]$$

- The new GetCells method, which supersedes GetLeafCells, also has additional options to control the output.

```
> r := GetCells(C, output=[samplepoints,descriptions]):
```

```
> r[1][1..5];
```

$$[[c=-1, b=0, x=-2], [c=-1, b=0, x=-1], [c=0, b=-1, x=-1], [c=0, b=-1, x=0], [c=0, b=0, x=-1]]$$

```
> r[2][1..5];
```

$$\left[c<0 \land x<\mathit{RootOf}\left(\_Z^2+b\_Z+c, \mathit{index}=\mathit{real}_1\right), c<0 \land x=\mathit{RootOf}\left(\_Z^2+b\_Z+c, \mathit{index}=\mathit{real}_1\right), c=0 \land b<0 \land x<\mathit{RootOf}\left(\_Z\left(\_Z+b\right), \mathit{index}=\mathit{real}_1\right), c=0 \land b<0 \land x=\mathit{RootOf}\left(\_Z\left(\_Z+b\right), \mathit{index}=\mathit{real}_1\right), c=0 \land b=0 \land x<0\right]$$

Various Small Improvements

- The ceil, floor, and round functions are 1.5-2x faster for some numeric arguments in Maple 2024.

```
> CodeTools:-Usage([seq(round(n), n=1..10^5)]):
```

```
memory used=85.45MiB, alloc change=0 bytes, cpu time=355.00ms, real
time=355.00ms, gc time=0ns
```

```
> CodeTools:-Usage([seq(floor(n/13), n=1..10^5)]):
```

```
memory used=191.96MiB, alloc change=0 bytes, cpu time=2.48s, real time=
1.78s, gc time=862.99ms
```

- The evalhf command uses a more accurate formula for computing the tangent, cotangent, hyperbolic tangent, and hyperbolic cotangent function in certain areas of the complex plane. As a consequence, evalhf is now used for computing these functions numerically in those parts of the complex plane whenever the requested precision is suitable for hardware float computation. This speeds the following computation up by 4-5x.

```
> CodeTools:-Usage([seq(tan(n * 1e-4 + 20.*I), n=1..10^4)]):
```

```
memory used=75.78MiB, alloc change=0 bytes, cpu time=2.49s, real time=
1.80s, gc time=846.68ms
```