Maple Quantum Chemistry Toolbox

The Maple Quantum Chemistry Toolbox from RDMChem is a powerful add-on product to Maple for computing and visualizing the electronic structure of molecules. In Maple 2026, QCT introduces significant new capabilities and enhancements that enable: (1) concise, index-based tensor algebra in Maple through the new TensorTools package, (2) object-oriented construction and manipulation of second-quantized operators through the new Operator framework, and (3) additional improvements across the Toolbox for computation, visualization, and workflow.

TensorTools: Index-Based Tensor Algebra

TensorTools brings concise, index-based tensor algebra to Maple. With *Einsum*, common contractions and index permutations can be expressed directly in Einstein notation, making interactive work with tensor expressions more transparent and less error-prone.

The following example converts a chemist-ordered 2-RDM to (ij|kl), contracts it to recover the 1-RDM, and verifies the result.

```
> with(QuantumChemistry):
  with(TensorTools):

  mol := MolecularGeometry("water"):
  out := CoupledCluster(mol, return_rdm="rdm1_and_rdm2"):

  N := round(add(out[mo_occ][])):

  D2 := Einsum("ikjl -> ijkl", out[rdm2]):          # reorder
  indices
  D1 := Einsum("ikjk -> ij", D2)/(N-1):             # partial trace

  Err  := convert(D1, Matrix) - convert(out[rdm1], Matrix):
  err2 := Einsum("ij,ij -> ", Err, Err):            # Frobenius norm
  squared
  printf("||D1(D2) - D1||_F^2 = %.3e\n", evalf(err2));
||D1(D2) - D1||_F^2 = 1.747e-16
```

Operator: Second-Quantized Algebra

Operator provides an object-oriented framework for second-quantized operators in Maple. With commands such as *Dagger*, *Reorder*, and *Combine*, users can build and manipulate fermionic expressions directly, automatically applying anticommutation relations and simplifying the result.

The following examples demonstrate normal ordering of fermionic operators, from a simple two-operator product to a Hamiltonian-like term.

```
> with(QuantumChemistry):
  ai := Operator("i"):
  aj := Operator("j"):
  cj := Dagger(aj):

  A := ai.cj:
  B := Combine(Reorder(A, [cj, ai]));
```

$$B := \delta_{j,i} - a_j^\dagger a_i$$

```
> ai := Operator("i"):
  aj := Operator("j"):
  al := Operator("l"):
  ak := Operator("k"):

  ci := Dagger(ai):
  cl := Dagger(al):

  H1 := h[i,k]*ci.ak + u[i,k,j,l]*ci.ak.cl.aj:
  H2 := Combine(Reorder(H1, [ci, cl, aj, ak]));
```

$$H2 := u_{i,k,j,l}\,\delta_{l,k}\,a_i^\dagger a_j + h_{i,k}\,a_i^\dagger a_k + u_{i,k,j,l}\,a_i^\dagger a_l^\dagger a_j a_k$$