

Tricks

1. Use `==` to test for equality. In effect, `A==B` is equivalent to `simplify(A-B)==0`.
2. In a script, line breaking is allowed where the scanner needs something to complete an expression. For example, the scanner will automatically go to the next line after an operator.
3. Setting `trace=1` in a script causes each line to be printed just before it is evaluated. Useful for debugging.
4. The last result is stored in symbol `last`.
5. Use `contract(A)` to get the mathematical trace of matrix A .
6. Use `binding(s)` to get the unevaluated binding of symbol s .
7. Use `s=quote(s)` to clear symbol s .
8. Use `float(pi)` to get the floating point value of π . Set `pi=float(pi)` to evaluate expressions with a numerical value for π . Set `pi=quote(pi)` to make π symbolic again.
9. Assign strings to unit names so they are printed normally. For example, setting `meter="meter"` causes symbol `meter` to be printed as meter instead of m_{eter} .
10. Use `expsin` and `expcos` instead of `sin` and `cos`. Trigonometric simplifications occur automatically when exponentials are used. See also `expform` for converting an expression to exponential form.
11. Use `rect(expform(f))` to maybe find a new form of trigonometric expression f .

```
f = cos(theta/2)^2
rect(expform(f))
```

$$\frac{1}{2} \cos(\theta) + \frac{1}{2}$$

12. The following exercise¹ demonstrates some `eval` tricks. Let

$$\psi = \frac{\phi_1 + \phi_2}{2} \exp\left(-\frac{iE_1 t}{\hbar}\right) + \frac{\phi_1 - \phi_2}{2} \exp\left(-\frac{iE_2 t}{\hbar}\right)$$

where ϕ_1 and ϕ_2 are orthogonal and operator A has the following eigenvalues.

$$A\phi_1 = a_1\phi_1$$

$$A\phi_2 = a_2\phi_2$$

¹See exercise 4-10 of *Quantum Mechanics* by Richard Fitzpatrick.

Verify that

$$\langle A \rangle = \int \psi^* A \psi dx = \frac{a_1 + a_2}{2} + \frac{a_1 - a_2}{2} \cos\left(\frac{(E_1 - E_2)t}{\hbar}\right)$$

Note: Because ϕ_1 and ϕ_2 are normalized we have $\int |\phi_1|^2 = \int |\phi_2|^2 = 1$. By orthogonality we have $\int \phi_1^* \phi_2 = 0$. Hence the integral can be accomplished with `eval`.

```
psi = (phi1 + phi2) / 2 exp(-i E1 t / hbar) +
      (phi1 - phi2) / 2 exp(-i E2 t / hbar)

Apsi = eval(psi, phi1, a1 phi1, phi2, a2 phi2) -- eigenvalues

phi1 = r1 exp(i theta1)
phi2 = r2 exp(i theta2)

A = conj(psi) Apsi

A = eval(A, r1^2, 1, r2^2, 1, r1 r2, 0) -- integral

A == (a1 + a2) / 2 + (a1 - a2) / 2 cos((E1 - E2) t / hbar)
```

13. Complex number functions `conj`, `mag`, etc. treat undefined symbols as representing real numbers. To define symbols that represent complex numbers, use separate symbols for the real and imaginary parts.

```
z = x + i y
conj(z) z

x^2 + y^2

z = A exp(i theta)
conj(z) z

A^2
```

14. Use `mag` for component magnitude, `abs` for vector magnitude.

```
y = (a, -b)
mag(y)

[a]
[b]

abs(y)

[a^2 + b^2]^(1/2)
```