Setup of the Algebra of Pauli Matrices in *Mathematica*

by Moritz Schubotz, José Luis Gómez-Muñoz, and Francisco Delgado-Cepeda
http://homepage.cem.itesm.mx/lgomez/quantum/
jose.luis.gomez@itesm.mx

Introduction

Pauli Algebra is implemented **only as an example** of how to define and use algebraic properties of operators in Quantum *Mathematica*. The standard *Mathematica* syntax and the special Quantum *Mathematica* syntax that are used are briefly explained. The user interested in Pauli Algebra for several spins (qubits) can use the Quantum`Computing` package, as shown in the following documents:

Mathematica file: http://homepage.cem.itesm.mx/lgomez/quantum/v7pauliope.nb

PDF: http://homepage.cem.itesm.mx/lgomez/quantum/v7pauliope.pdf

Load the Package

First load the Quantum`Notation` package. Write:
Needs["Quantum`Notation`"];
then press at the same time the keys SHIFT-ENTER to evaluate. *Mathematica* will load the package:

```
Needs["Quantum`Notation`"]
```

```
Quantum`Notation` Version 2.2.0. (July 2010)
A Mathematica package for Quantum calculations in Dirac bra-ket notation
by José Luis Gómez-Muñoz

Execute SetQuantumAliases[] in order to use
  the keyboard to enter quantum objects in Dirac's notation
SetQuantumAliases[] must be executed again in each new
  notebook that is created, only one time per notebook.
```

In order to use the keyboard to enter quantum objects write:
SetQuantumAliases[ ];
then press at the same time the keys SHIFT-ENTER to evaluate. The semicolon prevents *Mathematica* from printing the help message. Remember that SetQuantumAliases[ ] must be evaluated again in each new notebook:

```
SetQuantumAliases[];
```

Setup Pauli Algebra

The commands below define the Pauli Algebra, with σ_0 as the identity operator and σ_1, σ_2, σ_3 as the Pauli matrices. **The explanation is below, after the commands.**
Place the cursor on the definitions and press at the same time SHIFT-ENTER to evaluate, no output will be generated, but the definitions will be stored in the *Mathematica* session:

```
Needs["Quantum`Notation`"];
SetQuantumAliases[];
(* Important: Next code only works if
  Quantum`Computing` has NOT been loaded in the Mathematica session,
because Quantum`Computing` has its own defintions for the symbol σ *)

SetQuantumObject[σ];
(σa:0|1|2|3)^2 := σ0;
(σa:0|1|2|3)^† := σa;
σa:0|1|2|3 · σ0 := σa;
σ0 · σb:0|1|2|3 := σb;

σa:1|2|3 · σb:1|2|3 := KroneckerDelta[a, b] * σ0 + i * Sum[Signature[{a, b, c}] * σc, {c, 1, 3}];

[[σ0, σb:0|1|2|3]]_ := 0;

[[σa:1|2|3, σb:1|2|3]]_ := 2 * i * Sum[Signature[{a, b, c}] * σc, {c, 1, 3}];

[[σ0, σb:0|1|2|3]]+ := 2 * σb;
[[σa:1|2|3, σb:1|2|3]]+ := 2 * KroneckerDelta[a, b] * σ0;
```

There are several important details to notice in the definitions above:
Standard *Mathematica* keyboard aliases:
* The sigma σ can be entered by pressing the keys [ESC]s[ESC]
* The imaginary i can be entered by pressing the keys [ESC]ii[ESC]
Quantum keyboard aliases that can be used only if Needs["Quantum`Notation`"] and SetQuantumAliases[] have been executed in this notebook:
* Subscipts can be entered by pressing the keys [ESC]su[ESC]
* The noncommutative operator-application symbol · can be entered as [ESC]on[ESC]
* The sum template $\sum_{\square}^{\square}\square$ can be entered by pressing the keys [ESC]qs[ESC], as in "QuantumSum"
* Press [ESC]comm[ESC] for the commutator and [ESC]anti[ESC] for the anticommutator templates
Standard *Mathematica* syntax:
* The standard *Mathematica* command "Signature" plays the role of the Levi-Civita symbol
* The pattern **b**:0|1|2|3 means that the corresponding definition will be used only when **b** is equal to zero, one, two or three. On the other hand, the pattern **b**:1|2|3 means that the corresponding definition will be used only when **b** is equal to one, two or three.
* Notice the use of SetDelayed :=
* Do not type letters "O" or "o" when you mean number "0", *Mathematica* considers them as different symbols
Quantum *Mathematica* syntax that can be used only if Needs["Quantum`Notation`"] has been executed in this *Mathematica* session:
* SetQuantumObject[σ] makes σ a quantum object (operator)
* It is possible to assign values to powers of quantum objects. This is a modification of standard *Mathematica* behavior
* It is also possible to assign values to hermitian conjugates, noncommutative products, commutators and anticommutators of quantum objects.

Pauli Algebra calculations can be done after evaluation of the definitions above (place the cursor on the definitions above and press at the same time [SHIFT]-[ENTER])
It was defined that the square of any Pauli matrix is the identity:

```
(σ3)^2
```

σ_0

It was **not** defined what happens in the case of powers larger than 2, however Quantum *Mathematica* can obtain the correct answer from the definition for $(\sigma_a)^2$:

```
(σ3)^5
```

σ_3

This commutator is calculated using the **Signature[{a,b,c}]** command in the definition above. This command is equivalent to the Levi-Civita symbol if {a,b,c} are each one 1, 2 or 3

```
⟦σ1, σ3⟧_
```

$-2\,i\,\sigma_2$

This other commutator is calculated using the special definition for σ_0. Notice that the definition was made with σ_0 as first argument of the commutator, however it is used also if σ_0 is the second argument:

```
⟦σ1, σ0⟧_
```

0

This anticommutator is calculated using the **KroneckerDelta[a,b]** command in the definition above:

```
⟦σ3, σ3⟧+
```

$2\,\sigma_0$

This other anticommutator is calculated using the special definition for σ_0. Notice that the definition was made with σ_0 as first argument of the anticommutator, however it is used also if σ_0 is the second argument:

```
⟦σ1, σ0⟧+
```

$2\,\sigma_1$

Quantum expansions use the definitions made above:

```
Expand[(σ0 + i * σ1) · (σ0 + i * σ3) · (σ0 + i * σ2)]
```

$2\,i\,\sigma_1 + 2\,i\,\sigma_2$

Sample Application: Bloch Equations for Resonance Fluorescence

This is Moritz Schubotz's solution of exercises 3.10 from the textbook **QUANTUM MEASUREMENT AND CONTROL by H., Wiseman and G. Milburn 2010** www.cambridge.org/9780521804424

Every density matrix for a two level system can be written as a linear combination of Pauli-matrices, see next definition of ρ. Creation and annihilation operators σ_+ σ_- are also defined.

```
ρ = (σ0 + x[t] σ1 + y[t] σ2 + z[t] σ3) / 2
σ+ = (σ1 + I σ2) / 2
σ- = (σ+)†
```

$$\frac{1}{2}(\sigma_0 + \sigma_1\, x[t] + \sigma_2\, y[t] + \sigma_3\, z[t])$$

$$\frac{1}{2}(\sigma_1 + i\,\sigma_2)$$

$$\frac{1}{2}(\sigma_1 - i\,\sigma_2)$$

The effective Hamilton operator of a external driven system is given by equation (3.31) of Wiseman and Milburn, where Ω stands for the Rabi - frequency and Δ is the effective deturning of the atom:

```
H = Ω/2 σ1 + Δ/2 σ3
```

$$\frac{\Omega\,\sigma_1}{2} + \frac{\Delta\,\sigma_3}{2}$$

Furthermore the Lindblad super operator is given by equation (3.29) of Wiseman and Milburn. The evaluation of next definition does not produce any output, however it stores the definition of DD[]:

```
DD[A_] := Function[{ope}, A · ope · (A)† - ((A)† · A · ope + ope · (A)† · A)/2]
```

The Born - Markov Master Equation are given by equation (3.28) of Wiseman and Milburn. Here it is evaluated in the Hamiltonian that was previously defined:

```
dtρ = Simplify[Expand[ - i * [[H, ρ]]_ + γ * DD[σ-][ρ] ]];
TraditionalForm[dtρ]
```

$$\frac{1}{4}(\sigma_2\,(2\,\Delta\,x(t) - \gamma\,y(t) - 2\,\Omega\,z(t)) - \sigma_1\,(\gamma\,x(t) + 2\,\Delta\,y(t)) - 2\,\sigma_3\,(-\Omega\,y(t) + \gamma\,z(t) + \gamma))$$

Density matrices in Pauli basis, have the feature that the components x,y,z can be calculated by $x = \mathrm{Tr}(\sigma_1 \rho)$ etc. The same holds true for $d_t x$ etc. The Pauli – Matrices are all traceless with the exception of σ_0 where the trace is 2. So the equations of movement for x,z,t read:

```
Blochxyz = Simplify[Table[Expand[σk · dtρ] /. {σ1 → 0, σ2 → 0, σ3 → 0, σ0 → 2}, {k, 3}]];
TraditionalForm[MatrixForm[Blochxyz]]
```

$$\begin{pmatrix} \frac{1}{2}(-\gamma\, x(t) - 2\,\Delta\, y(t)) \\ \Delta\, x(t) - \frac{1}{2}\gamma\, y(t) - \Omega\, z(t) \\ \Omega\, y(t) - \gamma\,(z(t)+1) \end{pmatrix}$$

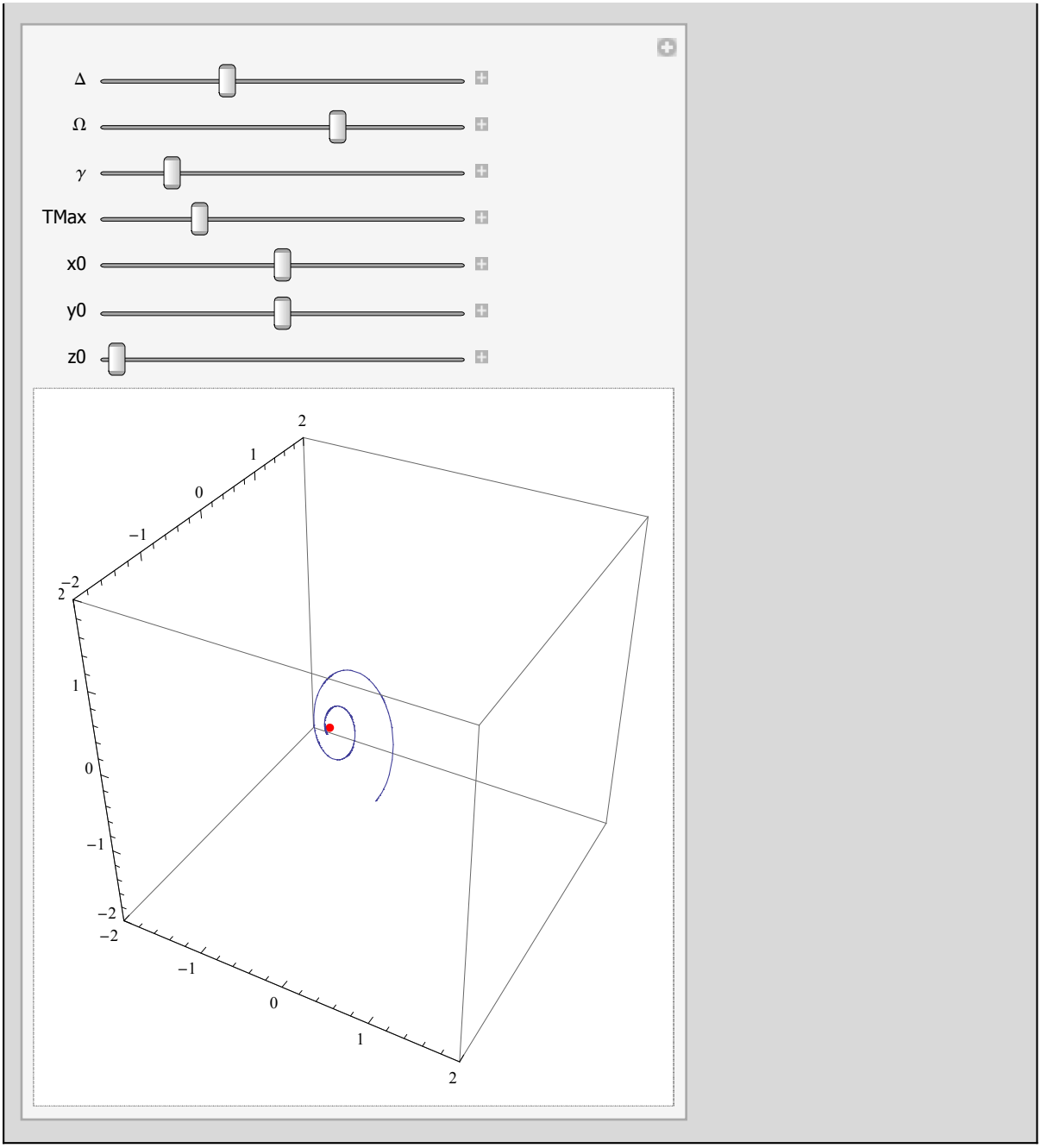
Furthermore the goal is to calculate a steady state for this equations.

```
X = {x[t], y[t], z[t]};
BlochStat[Δ_, Ω_, γ_] = Solve[Blochxyz == {0, 0, 0}, X];
X /. BlochStat[Δ, Ω, γ]
```

$$\left\{\left\{-\frac{4\,\Delta\,\Omega}{\gamma^2 + 4\,\Delta^2 + 2\,\Omega^2},\ \frac{2\,\gamma\,\Omega}{\gamma^2 + 4\,\Delta^2 + 2\,\Omega^2},\ -\frac{\gamma^2 + 4\,\Delta^2}{\gamma^2 + 4\,\Delta^2 + 2\,\Omega^2}\right\}\right\}$$

This can be visualized. The red dot is the steady state and the blue curve shows the movement of the density matrix in Bloch space for given initial conditions.

```
GLN[Δ_, Ω_, γ_] = Table[D[X[[k]], t] == Blochxyz[[k]], {k, 3}];
Manipulate[
 LSG = NDSolve[{GLN[Δ, Ω, γ], x[0] == x0, y[0] == y0, z[0] == z0}, X, {t, 0, TMax}]; P1 =
  ParametricPlot3D[X /. LSG, {t, 0, TMax}, PlotRange → {{-2, 2}, {-2, 2}, {-2, 2}}];
 P2 = Graphics3D[{Red, PointSize[Medium], Point[X /. BlochStat[Δ, Ω, γ]]}];
 Show[P1, P2],
 {{Δ, 1}, 0, 3}, {{Ω, 2}, 0, 3}, {{γ, 1 / 2}, 0, 3},
 {{TMax, 5}, 0, 20}, {{x0, 0}, -1, 1}, {{y0, 0}, -1, 1}, {{z0, -1}, -1, 1},
 SaveDefinitions → True
]
```



NOTE: Pauli Algebra is already implemented in the Quantum Computing package

This document shows how to implement Pauli Algebra **as an example** of the use of Quantum to create an "algebra". However, if you are interested in actually performing calculations with Pauli Algebra, you might want to use the implementation in the Quantum Computing package, where Pauli operators for different qubits (spins) can be used, it is possible to "expand" unitary operators in terms of Paul operators, it is possible to transform to matrix notation, among many other features. Here is the documentation for that implementation:

Mathematica file: http://homepage.cem.itesm.mx/lgomez/quantum/v7pauliope.nb

PDF: http://homepage.cem.itesm.mx/lgomez/quantum/v7pauliope.pdf

by Moritz Schubotz, José Luis Gómez-Muñoz, and Francisco Delgado-Cepeda
http://homepage.cem.itesm.mx/lgomez/quantum/
jose.luis.gomez@itesm.mx