Pauli Matrices as Unitary Gate Generators

> In section 2.2 of the text we introduced the three Pauli matrices

In[3]:=
```
σX = {{0, 1}, {1, 0}};
σY = {{0, -I}, {I, 0}};
σZ = {{1, 0}, {0, -1}};
```

> Which are typically expressed in the standard form

In[6]:=
```
MatrixForm[σX]
```

Out[6]//MatrixForm=

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

In[7]:=
```
MatrixForm[σY]
```

Out[7]//MatrixForm=

$$\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$$

In[8]:=
```
MatrixForm[σZ]
```

Out[8]//MatrixForm=

$$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

> Let's check if the commutation relations 2.25 are satisfied. Now

In[9]:=
```
σX.σY - σY.σX
```

Out[9]=
```
{{2 i, 0}, {0, -2 i}}
```

and

In[10]:=
```
2 I σZ
```

Out[10]=
```
{{2 i, 0}, {0, -2 i}}
```

> So obviously the first of these relations is satisfied. In Mathematica there is another way of testing an identity through the use of the logical equal command.

In[11]:=
```
? ==
```

lhs == *rhs* returns True if *lhs* and *rhs* are identical. ≫

Thus

In[12]:=
```
σX.σY - σY.σX == 2 I σZ
```

Out[12]=
```
True
```

Exercise: Explicitly evaluate the left and r.h.s. for the remaining expressions in (2.25) to prove the identities. Repeat using the Mathematica logical == command.

In (2.27) we gave an expression for a general 2×2 unitary matrix, and in the subsequent discussion illustrated how it can be expressed as a product of other unitary matrices (i.e 2.31, 2.32). Below we show, using the Mathematica `MatrixExp` command

In[13]:=
```
? MatrixExp
```

MatrixExp[m] gives the matrix exponential of m.
MatrixExp[m, v] gives the matrix exponential of m applied to the vector v. ≫

we can exponentiate a matrix without needing to evaluate an infinite series as in definition (2.28). For example, consider the operator

$U_X \equiv$ Exp[$i\,\alpha\,\sigma$X]

In[14]:=
```
Ux = MatrixExp[I α σX]
MatrixForm[Ux]
```

Out[14]=
```
{{Cos[α], i Sin[α]}, {i Sin[α], Cos[α]}}
```

Out[15]//MatrixForm=

$$\begin{pmatrix} \text{Cos}[\alpha] & i\,\text{Sin}[\alpha] \\ i\,\text{Sin}[\alpha] & \text{Cos}[\alpha] \end{pmatrix}$$

Which is the same expression derived in (2.31) by series expansion methods. In the same way,

In[16]:=
```
Uy = MatrixExp[I α σY]
MatrixForm[Uy]
```

Out[16]=
```
{{Cos[α], Sin[α]}, {-Sin[α], Cos[α]}}
```

Out[17]//MatrixForm=

$$\begin{pmatrix} \text{Cos}[\alpha] & \text{Sin}[\alpha] \\ -\text{Sin}[\alpha] & \text{Cos}[\alpha] \end{pmatrix}$$

In[18]:=
```
Uz = MatrixExp[I α σZ]
MatrixForm[Uz]
```

Out[18]=
```
{{ⅇ^(i α), 0}, {0, ⅇ^(-i α)}}
```

Out[19]//MatrixForm=

$$\begin{pmatrix} e^{i\alpha} & 0 \\ 0 & e^{-i\alpha} \end{pmatrix}$$

So let's now multiply, as in (2.33), the following three matrices

In[20]:=
```
Uproduct = MatrixExp[I (ϕ + β) / 2 σZ].MatrixExp[I θ σY].MatrixExp[ I (ϕ - β) / 2 σZ]
```

Out[20]=
$$\left\{\left\{e^{-\frac{1}{2} i (\beta-\phi)+\frac{1}{2} i (\beta+\phi)} \text{Cos}[\theta], e^{\frac{1}{2} i (\beta-\phi)+\frac{1}{2} i (\beta+\phi)} \text{Sin}[\theta]\right\},\right.$$
$$\left.\left\{-e^{-\frac{1}{2} i (\beta-\phi)-\frac{1}{2} i (\beta+\phi)} \text{Sin}[\theta], e^{\frac{1}{2} i (\beta-\phi)-\frac{1}{2} i (\beta+\phi)} \text{Cos}[\theta]\right\}\right\}$$

Ugh, what a mess!. Luckily there are the Mathematica commands

In[21]:=
```
?? Simplify
```

Simplify[*expr*] performs a sequence of algebraic and
other transformations on *expr* and returns the simplest form it finds.
Simplify[*expr*, *assum*] does simplification using assumptions. ≫

```
Attributes[Simplify] = {Protected}

Options[Simplify] =
 {Assumptions :→ $Assumptions, ComplexityFunction → Automatic, ExcludedForms → {},
  TimeConstraint → 300, TransformationFunctions → Automatic, Trig → True}
```

In[22]:=
```
?? FullSimplify
```

FullSimplify[*expr*] tries a wide range of transformations on *expr* involving
elementary and special functions and returns the simplest form it finds.
FullSimplify[*expr*, *assum*] does simplification using assumptions. ≫

```
Attributes[FullSimplify] = {Protected}

Options[FullSimplify] = {Assumptions :→ $Assumptions, ComplexityFunction → Automatic,
  ExcludedForms → {}, TimeConstraint → ∞, TransformationFunctions → Automatic, Trig → True}
```

In[23]:=
```
usimplify = FullSimplify[Uproduct]
MatrixForm[%]
```

Out[23]=
$$\left\{\left\{e^{i \phi} \text{Cos}[\theta], e^{i \beta} \text{Sin}[\theta]\right\}, \left\{-e^{-i \beta} \text{Sin}[\theta], e^{-i \phi} \text{Cos}[\theta]\right\}\right\}$$

Out[24]//MatrixForm=
$$\begin{pmatrix} e^{i \phi} \text{Cos}[\theta] & e^{i \beta} \text{Sin}[\theta] \\ -e^{-i \beta} \text{Sin}[\theta] & e^{-i \phi} \text{Cos}[\theta] \end{pmatrix}$$

and which agrees with expression (2.33) in the text.

Unitarity Properties

We claimed that the operator (2.33) is a unitary operator, which implies that a product with it's conjugate transpose (also called adjoint) should evaluate to an identity matrix. Let's check if that is

true. To obtain the conjugate transpose we use the Mathematica command

In[25]:=
```
?? ConjugateTranspose
```

ConjugateTranspose[*m*] or m† gives the conjugate transpose of m. ≫

```
Attributes[ConjugateTranspose] = {Protected}

ConjugateTranspose /: MakeBoxes[ConjugateTranspose[BoxForm`list_], TraditionalForm] /;
   BoxForm`sufficientVersionQ[6.1] :=
 TemplateBox[{Parenthesize[BoxForm`list, TraditionalForm, Power, Left]},
  ConjugateTranspose, SyntaxForm → SuperscriptBox]

ConjugateTranspose /: MakeBoxes[ConjugateTranspose[BoxForm`list_], TraditionalForm] :=
 SuperscriptBox[Parenthesize[BoxForm`list, TraditionalForm, Power, Left], †]

Options[ConjugateTranspose] = {AllowedHeads → Automatic}
```

In[26]:=
```
ConjugateTranspose[usimplify]
```

Out[26]=
$$\left\{\left\{e^{-i\,\text{Conjugate}[\phi]}\,\text{Conjugate}[\text{Cos}[\theta]],\ -e^{i\,\text{Conjugate}[\beta]}\,\text{Conjugate}[\text{Sin}[\theta]]\right\},\ \left\{e^{-i\,\text{Conjugate}[\beta]}\,\text{Conjugate}[\text{Sin}[\theta]],\ e^{i\,\text{Conjugate}[\phi]}\,\text{Conjugate}[\text{Cos}[\theta]]\right\}\right\}$$

When evaluating taking the conjugate of complex numbers Mathematica does not know which of the variables are real or complex. The command

In[27]:=
```
?? ComplexExpand
```

ComplexExpand[*expr*] expands *expr* assuming that all variables are real.
ComplexExpand[*expr*, {$x_1, x_2, \ldots$}] expands
expr assuming that variables matching any of the x_i are complex. ≫

```
Attributes[ComplexExpand] = {Protected}

Options[ComplexExpand] = {TargetFunctions → {Re, Im, Abs, Arg, Conjugate, Sign}}
```

allows one to expand the above expression so that

In[28]:=
```
udagger = FullSimplify[ComplexExpand[ConjugateTranspose[usimplify]]]
MatrixForm[udagger]
```

Out[28]=
$$\left\{\left\{e^{-i\,\phi}\,\text{Cos}[\theta],\ -e^{i\,\beta}\,\text{Sin}[\theta]\right\},\ \left\{e^{-i\,\beta}\,\text{Sin}[\theta],\ e^{i\,\phi}\,\text{Cos}[\theta]\right\}\right\}$$

Out[29]//MatrixForm=
$$\begin{pmatrix} e^{-i\,\phi}\,\text{Cos}[\theta] & -e^{i\,\beta}\,\text{Sin}[\theta] \\ e^{-i\,\beta}\,\text{Sin}[\theta] & e^{i\,\phi}\,\text{Cos}[\theta] \end{pmatrix}$$

is indeed the conjugate transpose of our operator. Now

In[30]:=
```
udagger.usimplify
Simplify[%]
MatrixForm[%]
```

Out[30]= $\{\{\text{Cos}[\theta]^2 + \text{Sin}[\theta]^2, 0\}, \{0, \text{Cos}[\theta]^2 + \text{Sin}[\theta]^2\}\}$

Out[31]= $\{\{1, 0\}, \{0, 1\}\}$

Out[32]//MatrixForm=

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

and so we unitarity is explicitly demonstrated.