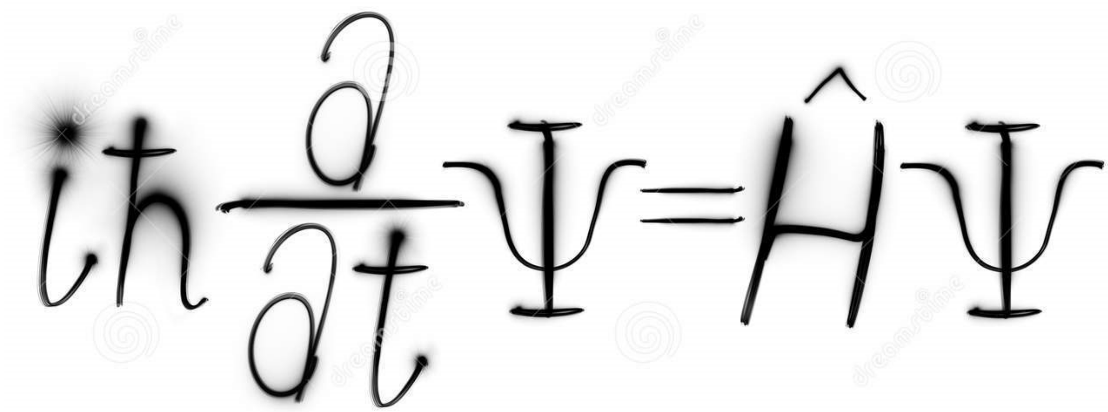
contact
name : Ahmed Moawad Gharib
email:Dr.gharaib85@gmail.com

Schrodinger equation

Out[1]=



Schrodinger equation for free particle easily solved symbolically with Mathematica::::::

$$\frac{\partial\,\psi[t]}{\partial t} = \frac{-i}{\hbar}\,\psi[t]$$
$$\psi[t] = c\,e^{-i\,h\,t}$$

Mathematica return solution with C[1] as initial condition

In[2]:=
```
DSolveValue[ψ'[t] == -ⅈ * h *  ψ[t], ψ[t], t]
```

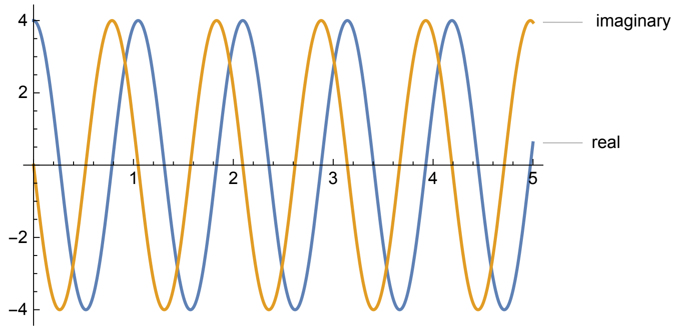
Out[2]= $e^{-i\,h\,t}\,C[1]$

if we specify initial condition and Hamiltonian

In[3]:=
```
initialvalue = 4;
h = 6;
DSolveValue[{ψ'[t] == -ⅈ * h *  ψ[t], ψ[0] == initialvalue}, ψ[t], t]
```

Out[5]= $4\,e^{-6\,i\,t}$

In[6]:=
```
Plot[4 Exp[-ⅈ 6 t] // ReIm // Evaluate, {t, 0, 5}, PlotLabels → {"real ", "imaginary"}]
```

Out[6]=


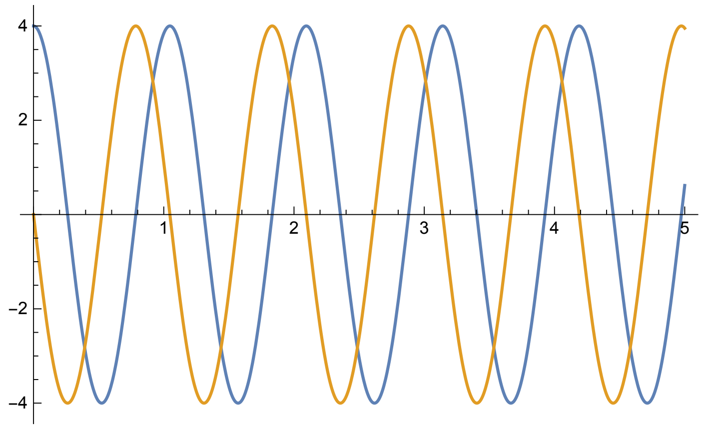
solve it numerically

NDSolveValue cant accept symbolics every thing is numeric ,initial condition must be known, also time domain must be known

In[7]:=
```
sol = NDSolveValue[{ψ'[t] == -ⅈ * 6 * ψ[t], ψ[0] == 4}, ψ, {t, 0, 5}]
```

Out[7]= InterpolatingFunction[Domain: {{0., 5.}} Output: scalar]

In[8]:=
```
Plot[sol[t] // ReIm // Evaluate, {t, 0, 5}]
```

Out[8]=


now its time to implement Schrodinger equation solver

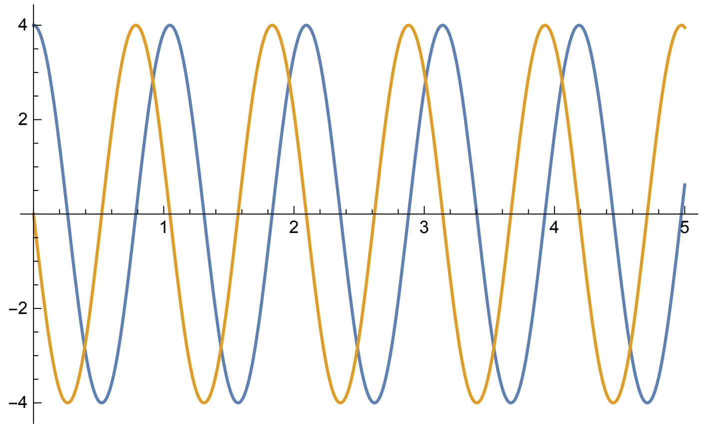
In[9]:=
```
ShrodingerSolve[h_, initialvalue_, tf_, t0_ : 0] :=
 Block[{ψ}, NDSolveValue[{ψ'[t] == -ⅈ * h * ψ[t], ψ[0] == initialvalue}, ψ, {t, t0, tf}]]
```

In[10]:=
```
newsol = ShrodingerSolve[6, 4, 5]
```

Out[10]= InterpolatingFunction[Domain: {{0., 5.}} Output: scalar]

In[11]:=
```
Plot[newsol[t] // ReIm // Evaluate, {t, 0, 5}]
```

Out[11]=



Mathematica so smart to understand compatibility with Hamiltonian And initial condition vector space and Bra Ket Dirac notation

In[12]:=
```
ShrodingerSolve[h_?NumberQ, initialvalue_?NumberQ, tf_, t0_:0] := Block[{ψ},
   NDSolveValue[{ψ'[t] == -ⅈ *h* ψ[t], ψ[0] == initialvalue}, ψ, {t, t0, tf}]];
(*___here h.ψ instead of h*ψ ___*)
ShrodingerSolve[h_?SquareMatrixQ, initialvalue_?ListQ, tf_, t0_:0] :=
  Block[{ψ}, NDSolveValue[{ψ'[t] == -ⅈ *h. ψ[t], ψ[0] == initialvalue}, ψ, {t, t0, tf}]];
```

lets solve first example see qutip-documentation p52

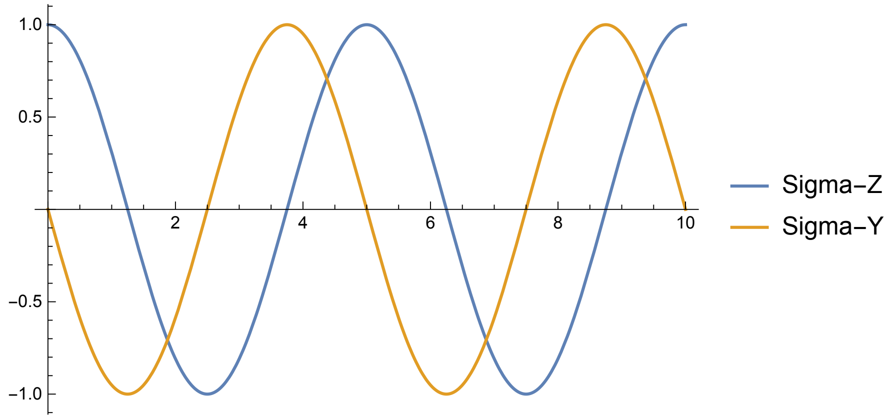
In[85]:=
```
H = 2 π * .1 * PauliMatrix[1];
ψ0 = {1, 0};
sol = ShrodingerSolve[H, ψ0, 10];
```

expectation value

<ψ*|o|ψ>

In[91]:=
```
Plot[{sol[t]*.PauliMatrix[3].sol[t], sol[t]*.PauliMatrix[2].sol[t]},
 {t, 0, 10}, PlotLegends → {"Sigma-Z", "Sigma-Y"}]
```

Out[91]=



In[92]:=
```
H = 2 π * .1 * PauliMatrix[1];
ψ0 = {{1}, {0}};
sol = ShrodingerSolve[H, ψ0, 10];
```

expectation value same as above but insted od conjugate we use conjugate transpost

but also

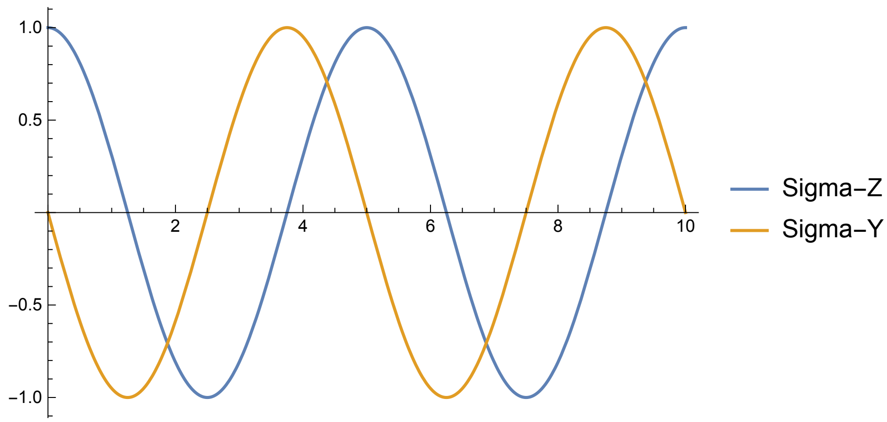
we we go beyound and use another forn

Tr[o.ρ]

In[102]:=
```
rho[t_] := sol[t].sol[t]†
```

In[108]:=
```
Plot[{Tr[rho[t].PauliMatrix[3]], Tr[rho[t].PauliMatrix[2]]},
 {t, 0, 10}, PlotLegends → {"Sigma-Z", "Sigma-Y"}]
```

Out[108]=



In[101]:=
```
sol[1].sol[1]†
```

Out[101]=
```
{{0.654508 + 0. i, 0. + 0.475528 i}, {0. - 0.475528 i, 0.345491 + 0. i}}
```