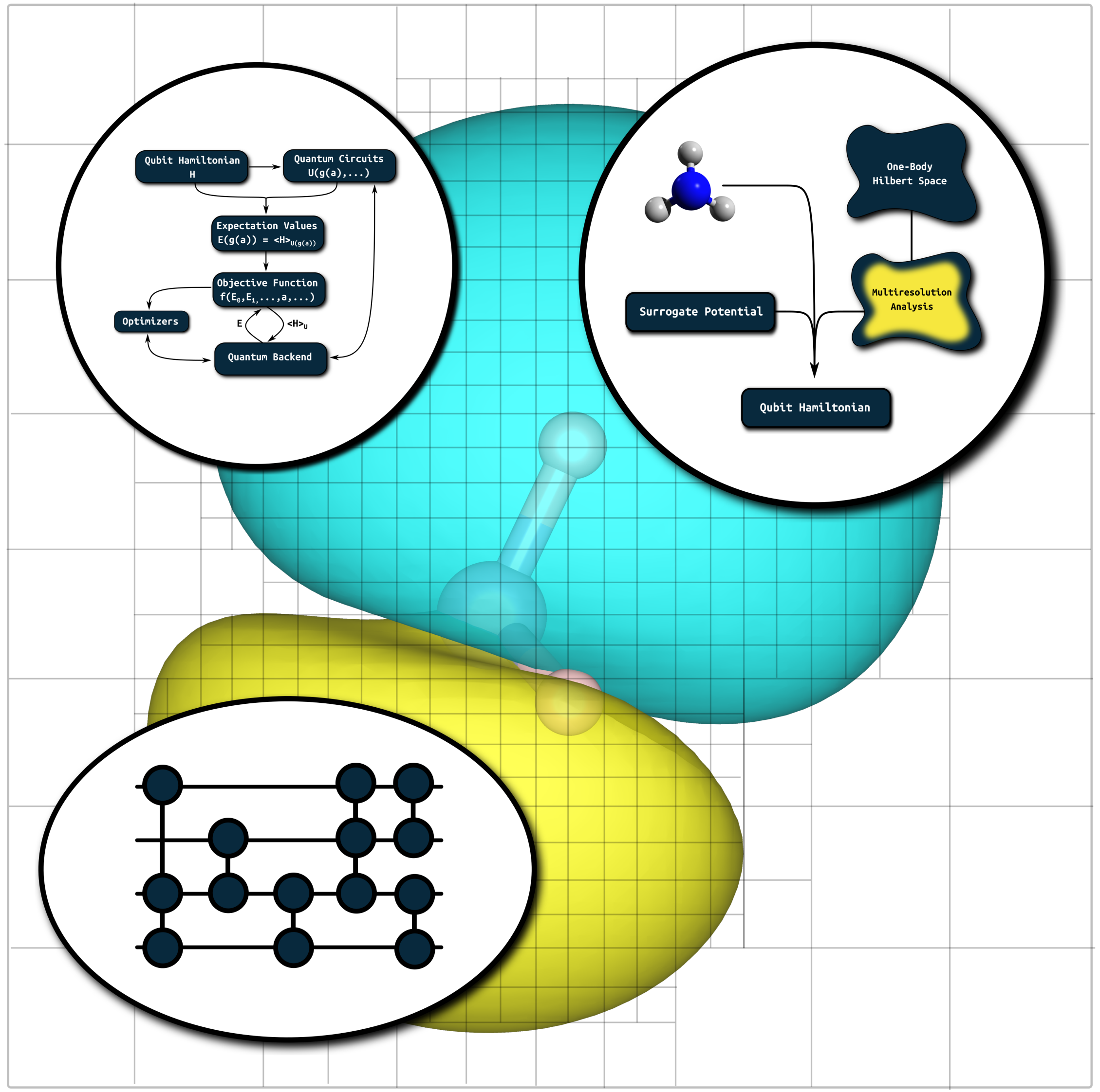
Getting Started with Tequila

Jakob S. Kottmann

University of Toronto

@JakobKottmann

github/tequilahub



```
# install from PyPi
pip install tequila-basic

# install from github
pip install git+https://github.com/tequilahub/tequila.git

# install with windows (not recommended)
pip install https://github.com/tequilahub/tequila@windows

# recommended: install fast backend
pip install qulacs
```

```
Find Slides here:
https://github.com/kottmanj/talks_and_material/

Will also contain small scripts that reproduce data shown in the talk

Problems/Wishes/Feedback:
firstname.lastname@gmail.com
twitter pm: @jakobkottmann
```

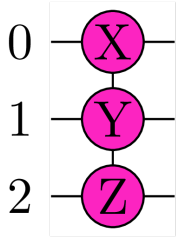
Circuits and Notation

$$e^{-i\frac{a}{2}X(0)Y(1)Z(2)}$$

```
U=tq.gates.ExpPauli(paulistring="X(0)Y(1)Z(2)", angle="a")
```

magenta: parametrized gate

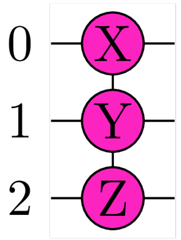
$$e^{-i\frac{a}{2}X(0)Y(1)Z(2)} =$$



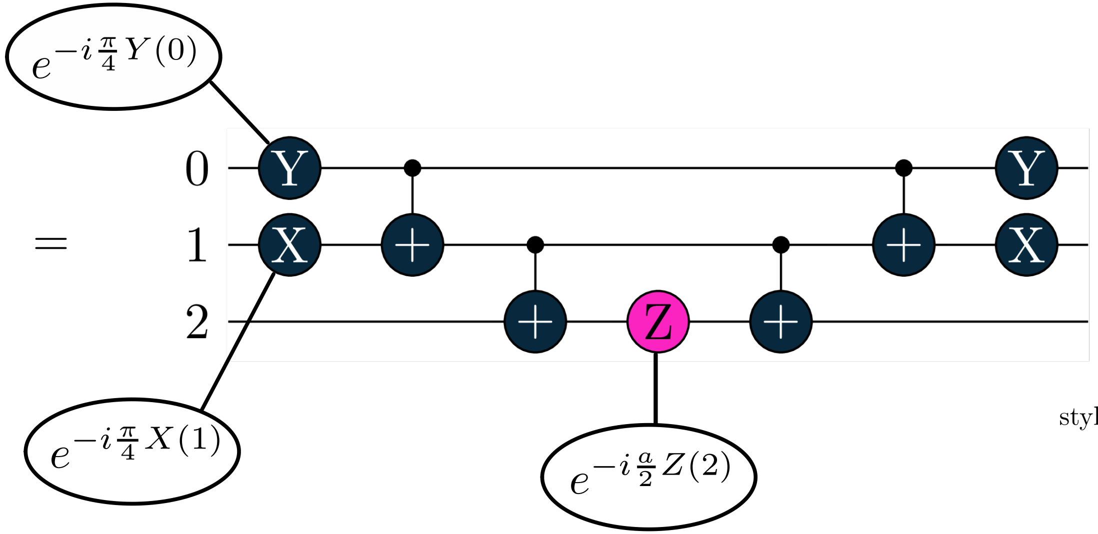
```
U=tq.gates.ExpPauli(paulistring="X(0)Y(1)Z(2)", angle="a")
```

magenta: parametrized gate

$$e^{-i\frac{a}{2}X(0)Y(1)Z(2)} =$$



$$=$$

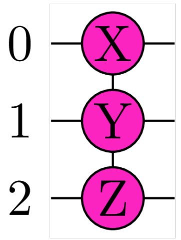


style=tequila

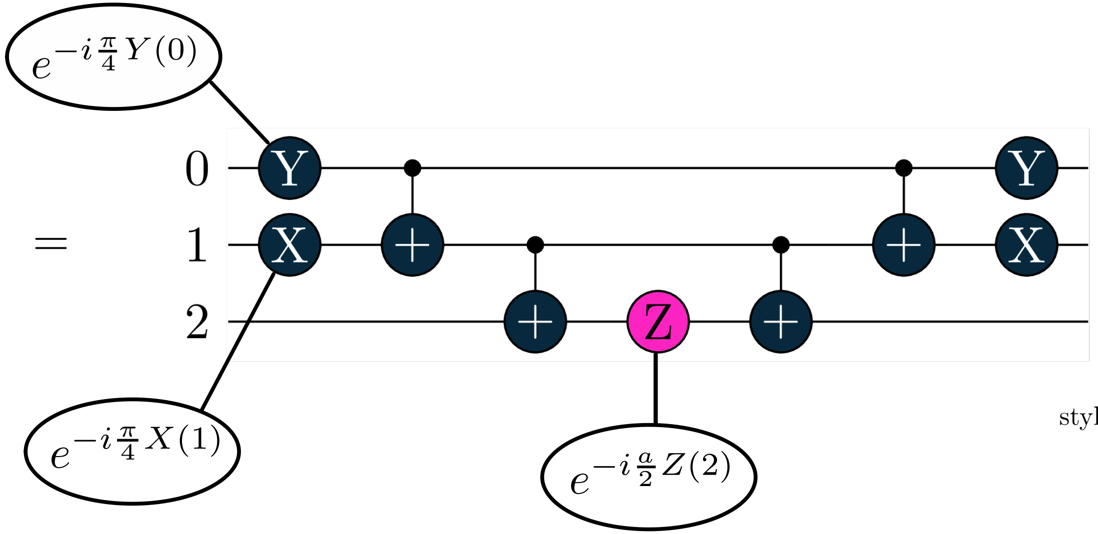
```
U=tq.gates.ExpPauli(paulistring="X(0)Y(1)Z(2)", angle="a")
```

magenta: parametrized gate

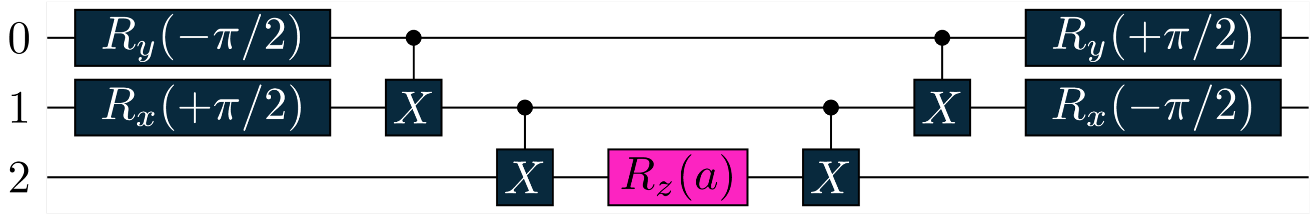
$$e^{-i\frac{a}{2}X(0)Y(1)Z(2)} =$$



$=$



style=tequila



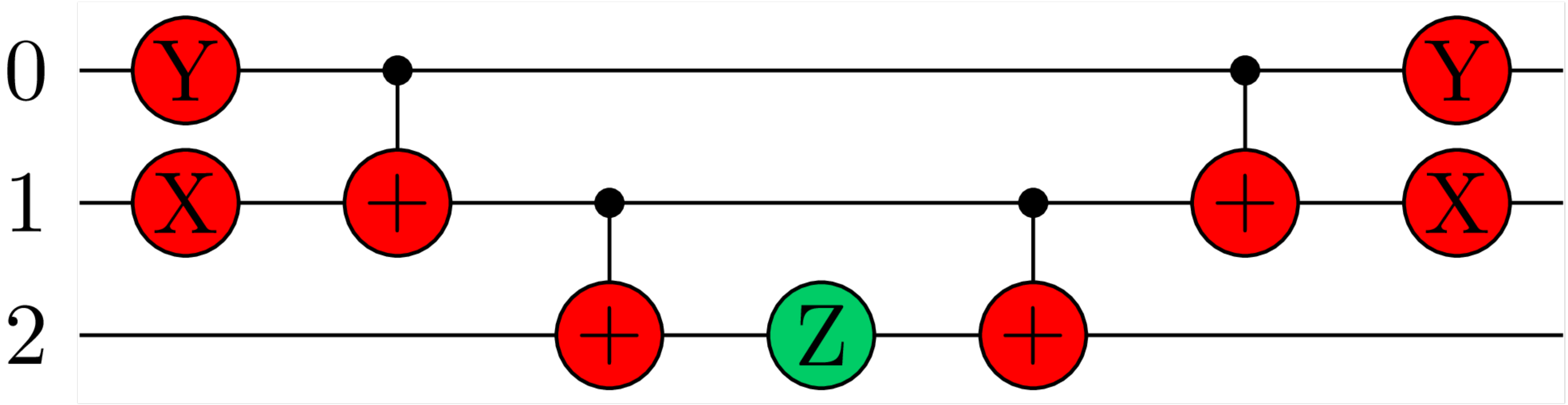
style=standard

create plots: (png, pdf, tex, qpic)

```
U = tq.compile_circuit(U)
U.export_to("filename1.png", style="tequila")
U.export_to("filename2.png", style="standard")
U.export_to("filename3.png", style="plain")
```

needs qpic and pdf-to-png converter → see qpic docs

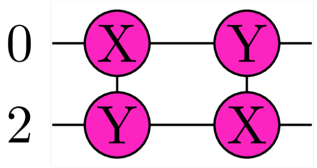
```
U.export_to(filename="asd.png", style="tequila", gatecolor1="red", textcolor1="black", gatecolor2="mine", colors=[{"name":"mine", "rgb":(0.0, 0.8, 0.4)}])
```



produces qpic file ("asd.qpic") that can also be manipulated manually

```
linux: pip install qpic # is sufficient when pdflatex is there
mac: pip install qpic # needs imagemagick installed for pngs
windows: no qpic (sorry). Can still produce qpic files (filename="whatever.qpic) and convert in virtual machine

see github.com/qpic for more

not happy with qpic? Create your own function to export circuits and become a contributor :-)
```

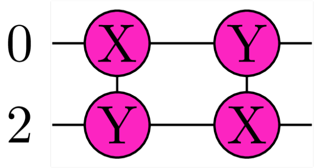
```
U2 = tq.gates.QubitExcitation(target=[0,2], angle="a")
```



$$e^{-i\frac{a}{2}(\sigma_+(0)\sigma_-(2)+\sigma_-(0)\sigma_+(2))}$$

$$\sigma_\pm = \frac{1}{2}(X \pm Y)$$

```
U2 = tq.gates.QubitExcitation(target=[0,2], angle="a")
```

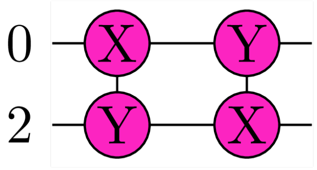


$$e^{-i\frac{a}{2}(\sigma_+(0)\sigma_-(2)+\sigma_-(0)\sigma_+(2))}$$

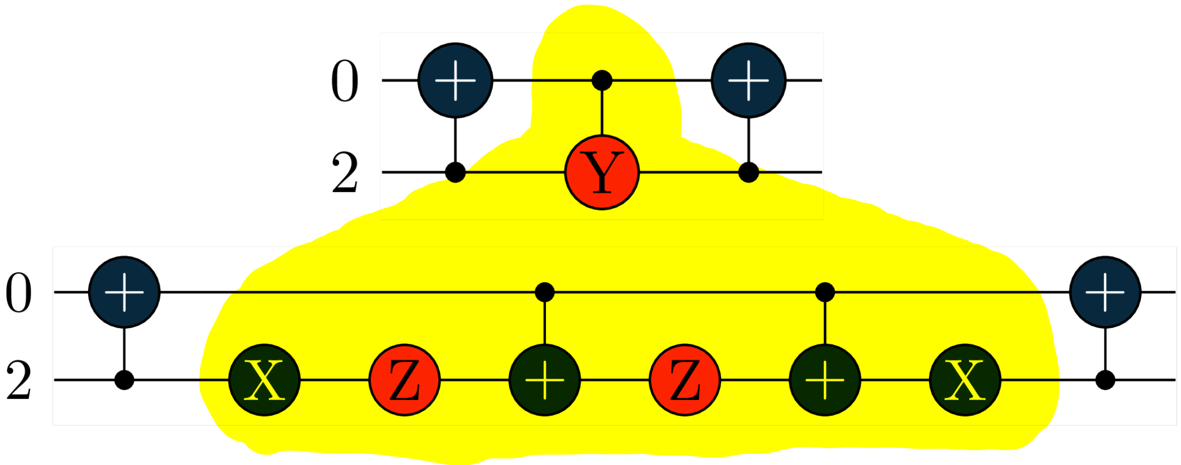
$$\sigma_\pm = \frac{1}{2}(X \pm Y)$$

```
U = tq.gates.X(0) + U2
U = tq.compile(U, backend=...)
U(variables={"a":1.0})
>>> +0.8776|100> +0.4794|001>
```

```
U2 = tq.gates.QubitExcitation(target=[0,2], angle="a")
```



$$=$$



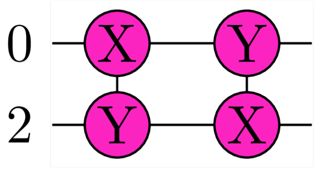
$$e^{-i\frac{a}{2}(\sigma_+(0)\sigma_-(2)+\sigma_-(0)\sigma_+(2))} \quad =$$

$$\sigma_\pm = \frac{1}{2}(X \pm Y)$$

```
U = tq.gates.X(0) + U2
U = tq.compile(U, backend=...)
U(variables={"a":1.0})
>>> +0.8776|100> +0.4794|001>
```

automatic translation depending on backend

```
U2 = tq.gates.QubitExcitation(target=[0,2], angle="a")
```

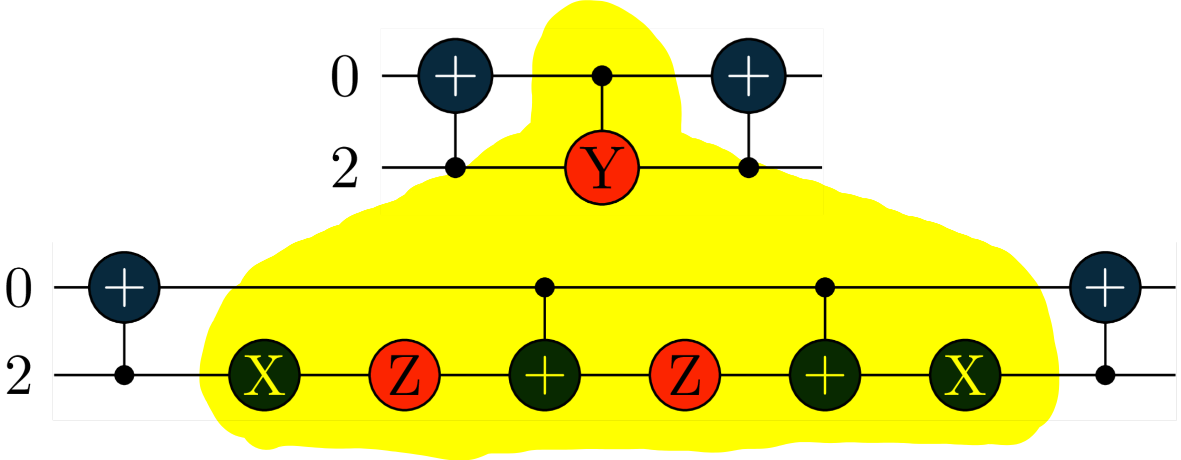


=

$$e^{-i\frac{a}{2}(\sigma_+(0)\sigma_-(2)+\sigma_-(0)\sigma_+(2))}$$

$$\sigma_\pm = \frac{1}{2}(X \pm Y)$$

=




```
U = tq.gates.X(0) + U2
U = tq.compile(U, backend=...)
U(variables={"a":1.0})
>>> +0.8776|100> +0.4794|001>
```

automatic translation depending on backend

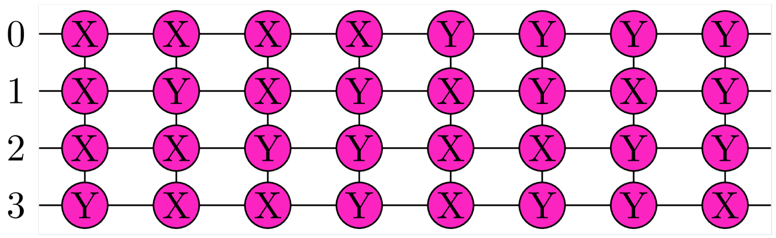
manual:

```
# default: lowest level
U1 = tq.compile_circuit(U)

# others
U1 = tq.compile_circuit(U, exponential_pauli=False)
U1 = tq.compile_circuit(U, controlled_rotation=False)
```

```
U4 = tq.gates.QubitExcitation(target=[0,2,1,3], angle="a")
```

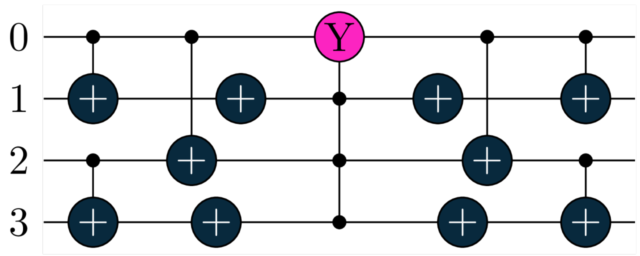
=

backend=qulacs




```
U = tq.gates.X([0,1]) + U4
U = tq.compile(U, backend=...)
U(variables={"a":1.0})
>>> +0.8776|1100> +0.4794|0011>
```

backend=cirq

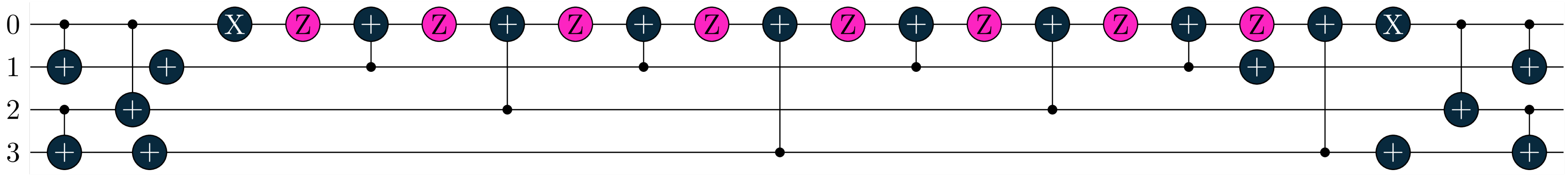


Yordanov *et.al.* arxiv:2005.14475

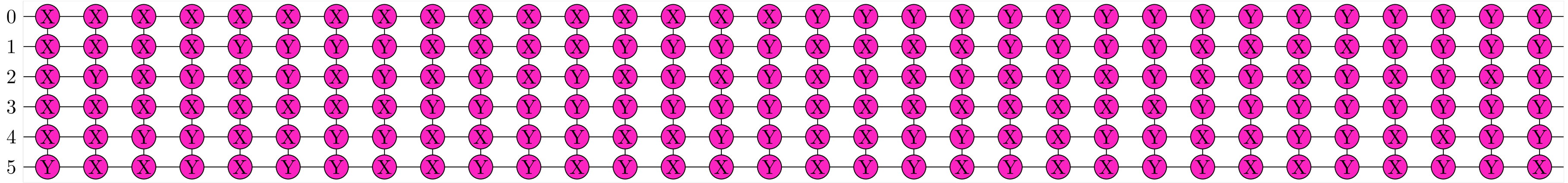
simila: Anselmetti *et.al.* arxiv:2104.05695

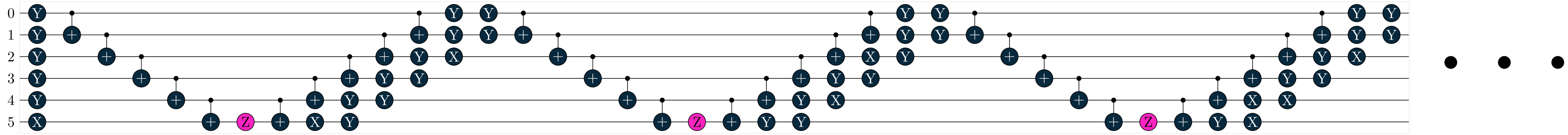
multicontrol compiling: G. Tsilimigkounakis (qosf project)

backend=qiskit



```
U4 = tq.gates.QubitExcitation(target=[0,3,1,4,2,5], angle="a")
```





Triple and higher qubit excitations:

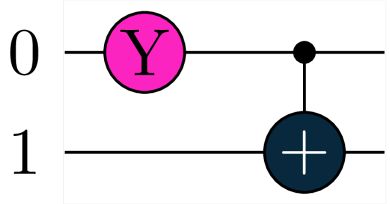
No optimization yet .... (should work analogue to single and double qubit excitations)

```
Hamiltonian, ExpectationValue
```

$$\langle\psi|H|\psi\rangle \equiv \langle 0|U^\dagger H U|0\rangle \equiv \langle H\rangle_U$$

$$\langle\psi|H|\psi\rangle \equiv \langle 0|U^\dagger H U|0\rangle \equiv \langle H\rangle_U$$

example: $H = X(0)Y(1) + \frac{1}{2}Z(0)$

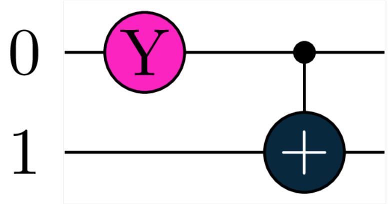
$U =$ 


```
from tequila.paulis import X,Y,Z
from tequila.gates import Ry, CNOT

H = X(0)*Y(1) + 0.5*Z(0)
U = Ry(angle="a", target=0)
U+= CNOT(0,1)

E = tq.ExpectationValue(H=H, U=U)

f = tq.compile(E)
```

$$\langle\psi|H|\psi\rangle \equiv \langle 0|U^\dagger H U|0\rangle \equiv \langle H\rangle_U$$

example: $H = X(0)Y(1) + \frac{1}{2}Z(0)$

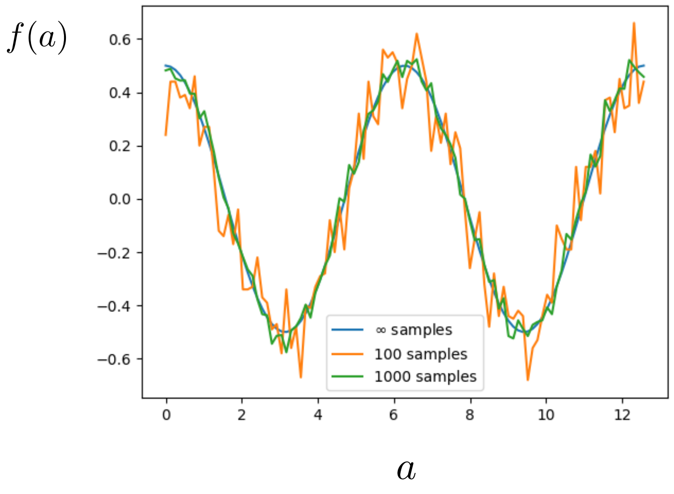
$U =$




```
from tequila.paulis import X,Y,Z
from tequila.gates import Ry, CNOT

H = X(0)*Y(1) + 0.5*Z(0)
U = Ry(angle="a", target=0)
U+= CNOT(0,1)

E = tq.ExpectationValue(H=H, U=U)

f = tq.compile(E)
```

corresponds to ∞ samples. Perfect simulation

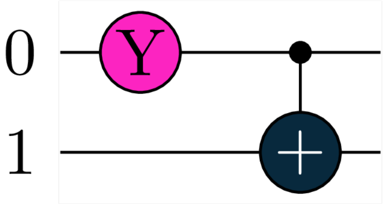
```
f(variables={"a":1.0})
```



device noise etc .... see tutorials (Sumner Alperin-Lea)

$$\langle\psi|H|\psi\rangle \equiv \langle 0|U^{\dagger}HU|0\rangle \equiv \langle H\rangle_U$$

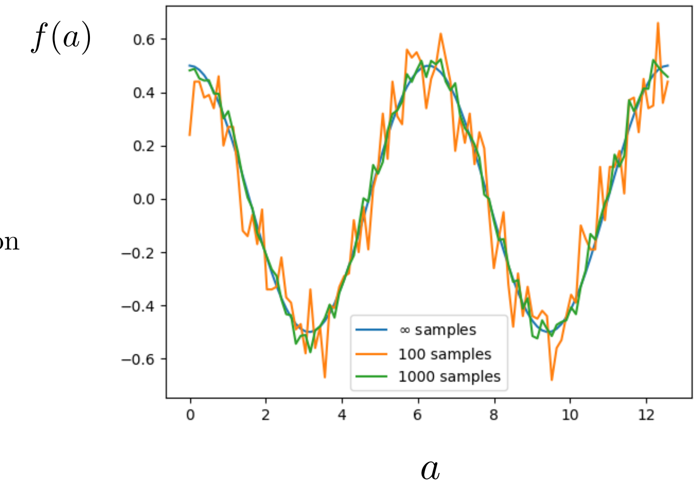
example: $H = X(0)Y(1) + \frac{1}{2}Z(0)$

$U =$ 

```
from tequila.paulis import X,Y,Z
from tequila.gates import Ry, CNOT

H = X(0)*Y(1) + 0.5*Z(0)
U = Ry(angle="a", target=0)
U+= CNOT(0,1)

E = tq.ExpectationValue(H=H, U=U)

f = tq.compile(E)
```

corresponds to ∞ samples. Perfect simulation

```
f(variables={"a":1.0})
```



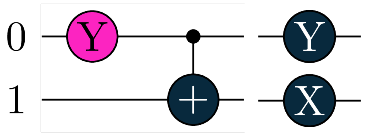

```
f(variables={"a":1.0}, samples=100)
```

$\langle X(0)Y(0)\rangle_U$

shift into XY basis

measure both qubits 100 times

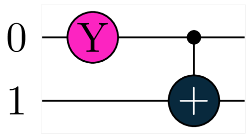
accumulate expectation value



$\langle Z(0)\rangle_U$

measure qubit 0 100 times

accumulate expectation value



device noise etc .... see tutorials (Sumner Alperin-Lea)

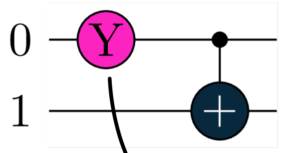
example_expectationvalue.py

$$L(a) = \langle H \rangle_{U(a)} + e^{-\left(\frac{\partial}{\partial a}\langle H \rangle_{U(a)}\right)^2}$$

how does this function look?

$$H = X(0)X(1) + \frac{1}{2}Z(0) + Y(1)$$

$U =$



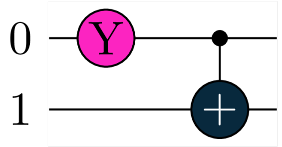
$$e^{-i\frac{f(a)}{2}Y(0)}$$

$$f(a) = e^{-a^2}$$

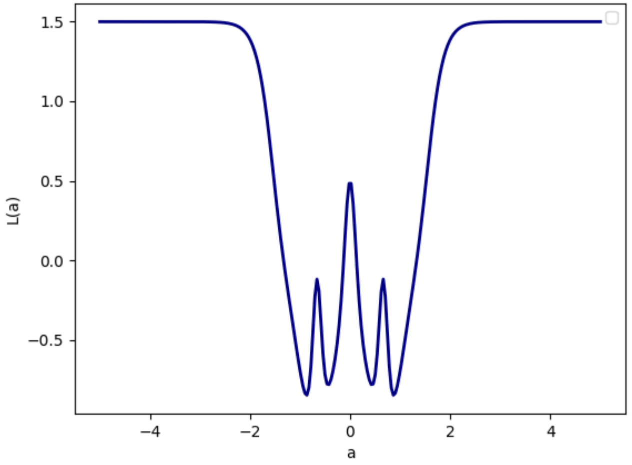
```
a = tq.Variable("a")
f = (-a**2).apply(tq.numpy.exp)

U = tq.gates.Ry(angle=f*numpy.pi, target=0)
U += tq.gates.CNOT(0,1)

H = tq.paulis.from_string("-1.0*X(0)X(1)+0.5*Z(0)+Y(1)")

E = tq.ExpectationValue(H=H, U=U)
dE = tq.grad(E, "a")

L = E + (-dE**2).apply(tq.numpy.exp)
```

$$L(a) = \langle H \rangle_{U(a)} + e^{-\left(\frac{\partial}{\partial a}\langle H \rangle_{U(a)}\right)^2}$$

$$H = X(0)X(1) + \frac{1}{2}Z(0) + Y(1)$$

$U =$ 

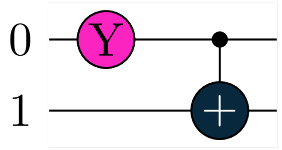

```
tq.compile(L)
```




```
a = tq.Variable("a")
f = (-a**2).apply(tq.numpy.exp)

U = tq.gates.Ry(angle=f*numpy.pi, target=0)
U += tq.gates.CNOT(0,1)

H = tq.paulis.from_string("-1.0*X(0)X(1)+0.5*Z(0)+Y(1)")

E = tq.ExpectationValue(H=H, U=U)
dE = tq.grad(E, "a")

L = E + (-dE**2).apply(tq.numpy.exp)
```

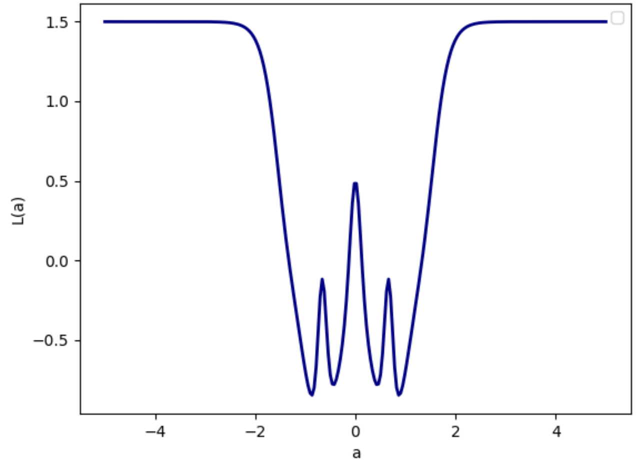
$$L(a) = \langle H \rangle_{U(a)} + e^{-\left(\frac{\partial}{\partial a}\langle H \rangle_{U(a)}\right)^2}$$

$$H = X(0)X(1) + \frac{1}{2}Z(0) + Y(1)$$

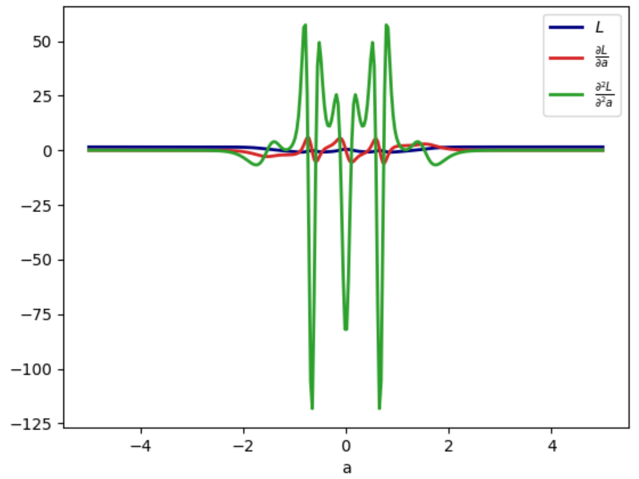
$U =$ 


```
a = tq.Variable("a")
f = (-a**2).apply(tq.numpy.exp)

U = tq.gates.Ry(angle=f*numpy.pi, target=0)
U += tq.gates.CNOT(0,1)

H = tq.paulis.from_string("-1.0*X(0)X(1)+0.5*Z(0)+Y(1)")

E = tq.ExpectationValue(H=H, U=U)
dE = tq.grad(E, "a")

L = E + (-dE**2).apply(tq.numpy.exp)
```

```
tq.compile(L)
```




```
dL = tq.grad(L, "a")
dL2 = tq.grad(dL, "a")
```



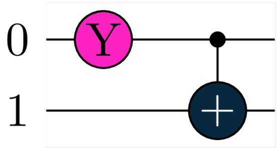

```
print(L)
```

```
>>> Objective with 3 unique expectation values
    total measurements = 9
    variables          = [a]
    types              = not compiled
```

example_objective.py

$$L(a) = \langle H \rangle_{U(a)} + e^{-\left(\frac{\partial}{\partial a}\langle H \rangle_{U(a)}\right)^2}$$

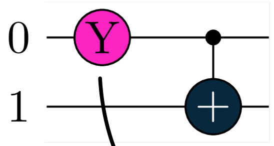
$$H = X(0)X(1) + \frac{1}{2}Z(0) + Y(1)$$

$U =$ 

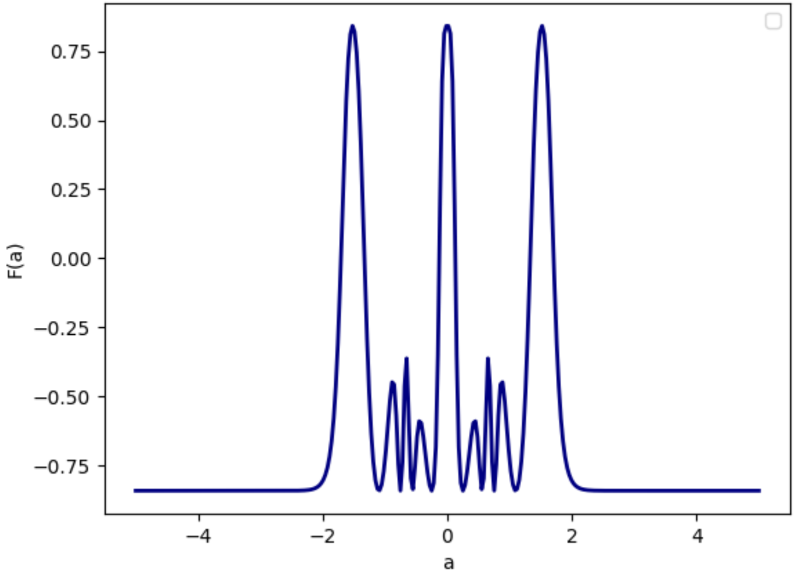

```
L = tq.compile(L)
U2 = tq.gates.Ry(angle=L, target=0)
U2+= tq.gates.CNOT(0,1)
```

$$F(a) = \sin\left(\langle H_2 \rangle_{U_2}\right)$$

$$H_2 = X(0) + X(1) + X(0)X(1)$$

$U_2 =$ 

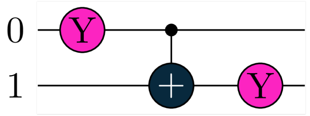
$$e^{-i\frac{L(a)}{2}Y(0)}$$



example_nested_objective.py

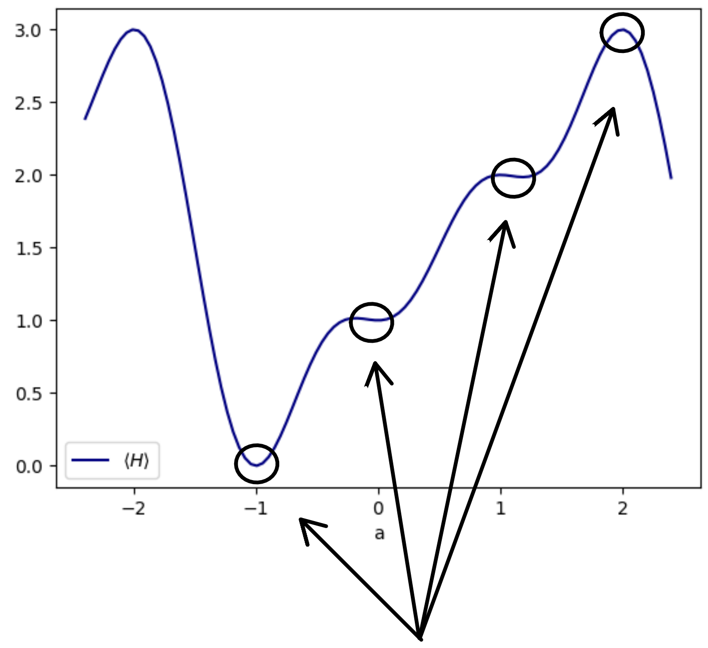
Example: VQE

$$H = \frac{3}{2} - \frac{1}{2}\left(Z(1) - Z(0) + Z(0)Z(1) + X(1) - Z(0)X(1)\right)$$

$U =$ 

$$\min \langle H \rangle_{U(a,b)}$$

Task: Prepare Ground State



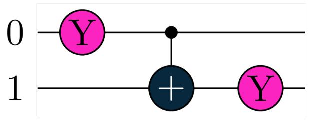
eigenstates

Variational Quantum Eigensolver (VQE):

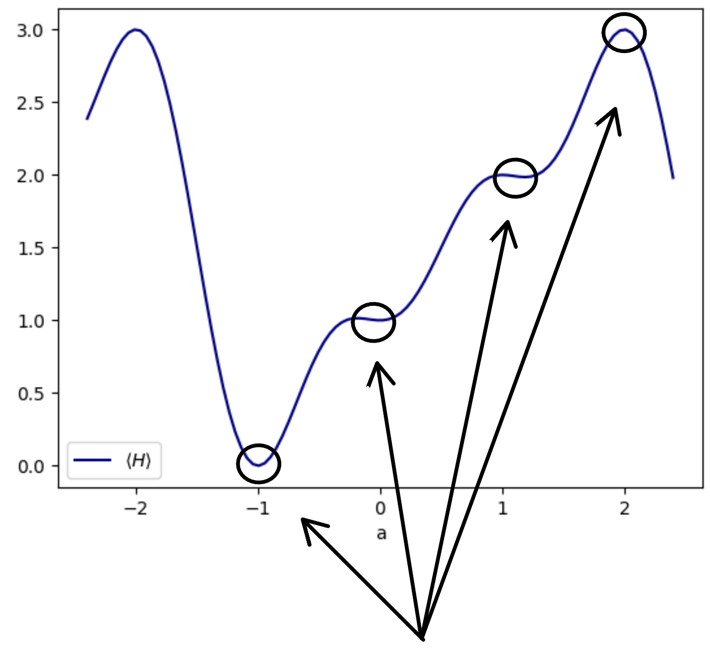
Peruzzo, McClean *et.al*. Nat. Comm. 2014

McClean *et.al* NJP, 2016

$$H = \frac{3}{2} - \frac{1}{2}\left(Z(1) - Z(0) + Z(0)Z(1) + X(1) - Z(0)X(1)\right)$$

$U =$ 

$$\min \langle H \rangle_{U(a,b)}$$



Task: Prepare Ground State

```
import tequila as tq
from tequila.hamiltonian.paulis import X,Y,Z
import numpy

H = 1.5-0.5*(Z(1)-Z(0)+Z(0)*Z(1)+X(1)-Z(0)*X(1))

a = tq.Variable("a")
U = tq.gates.Ry(angle=a*numpy.pi,target=0)
U+= tq.gates.CNOT(0,1)
U+= tq.gates.Ry(angle=(a/2)*numpy.pi, target=1)

E = tq.ExpectationValue(H=H, U=U)

result = tq.minimize(E, initial_values="random")
```

Sumner Alperin-Lea :

Interfaces to SciPy (default), GPyOpt, Phoenics, Adam etc

more on optimizers: github/tequilahub/tequila-tutorials/

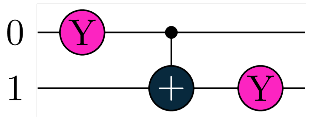
vqe.py

Variational Quantum Eigensolver (VQE):

Peruzzo, McClean *et.al*. Nat. Comm. 2014

McClean *et.al* NJP, 2016

$$H = \frac{3}{2} - \frac{1}{2}\left(Z(1) - Z(0) + Z(0)Z(1) + X(1) - Z(0)X(1)\right)$$

$U =$ 

Task: Prepare Ground State

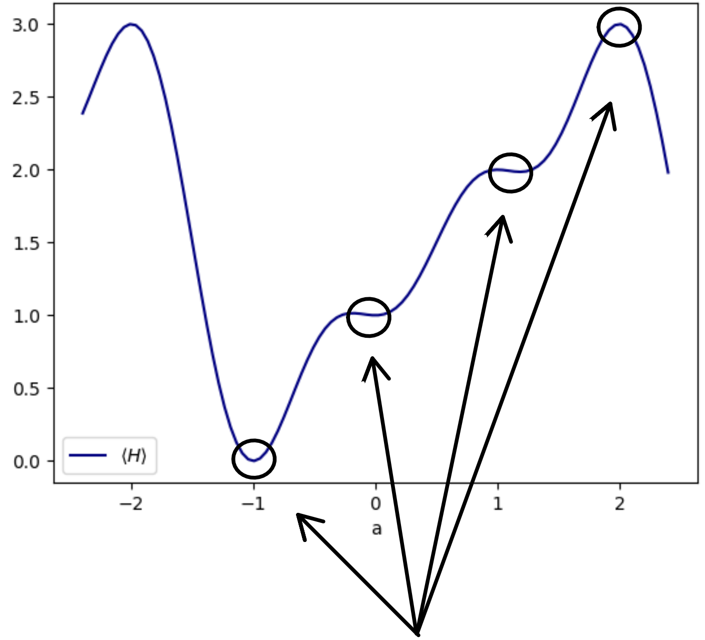
```
import tequila as tq
from tequila.hamiltonian.paulis import X,Y,Z
import numpy

H = 1.5-0.5*(Z(1)-Z(0)+Z(0)*Z(1)+X(1)-Z(0)*X(1))

a = tq.Variable("a")
U = tq.gates.Ry(angle=a*numpy.pi,target=0)
U+= tq.gates.CNOT(0,1)
U+= tq.gates.Ry(angle=(a/2)*numpy.pi, target=1)

E = tq.ExpectationValue(H=H, U=U)

result = tq.minimize(E, initial_values="random")
```

Sumner Alperin-Lea :

Interfaces to SciPy (default), GPyOpt, Phoenics, Adam etc

more on optimizers: github/tequilahub/tequila-tutorials/

vqe_excited_state.py



eigenstates

useful in daily life:

```
v,vv = numpy.linalg.eigh(H.to_matrix())

for i in range(len(v)):
    wfn=tq.QubitWaveFunction(vv[:,i])
    print("E{}={:+2.1f}, wfn=".format(i,v[i]), wfn)
```

↓

```
>>>>E0=+0.0, wfn= -0.7071|10> -0.7071|11>
    E1=+1.0, wfn= +1.0000|00>
    E2=+2.0, wfn= -0.7071|10> +0.7071|11>
    E3=+3.0, wfn= +1.0000|01>
```

Variational Quantum Eigensolver (VQE):

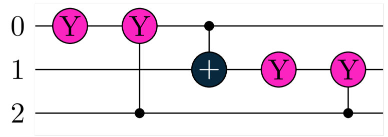
Peruzzo, McClean *et.al.* Nat. Comm. 2014

McClean *et.al* NJP, 2016

Example: Circuit Optimization

$$H = \frac{3}{2} - \frac{1}{2}\left(Z(1) - Z(0) + Z(0)Z(1) + X(1) - Z(0)X(1)\right)$$

eigenvalues: $\{0, 1, 2, 3\}$

$U =$ 

Task: Prepare eigenstates of H depending on qubit 2

qubit 2 in $|0\rangle$: Prepare ground state

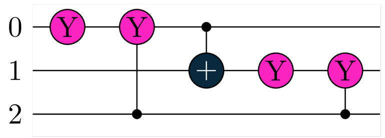
qubit 2 in $|1\rangle$: Prepare highest eigenstate

state_preparation_hamiltonian.py
state_preparation_fidelity.py

$$H = \frac{3}{2} - \frac{1}{2}\left(Z(1) - Z(0) + Z(0)Z(1) + X(1) - Z(0)X(1)\right)$$

eigenvalues: $\{0, 1, 2, 3\}$

$U =$



Task: Prepare eigenstates of H depending on qubit 2

qubit 2 in $|0\rangle$: Prepare ground state

qubit 2 in $|1\rangle$: Prepare highest eigenstate

Two Strategies:

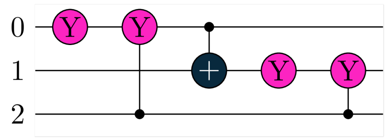
$$\min \; \left(\langle H\rangle_U - \langle H\rangle_{XU}\right)$$

$$\text{or:} \;\; \min \; \left(\langle P_{|000\rangle}\rangle_{UU_0^\dagger} + \langle P_{|001\rangle}\rangle_{XUU_1^\dagger}\right)$$

state_preparation_hamiltonian.py
state_preparation_fidelity.py

$$H = \frac{3}{2} - \frac{1}{2}\left(Z(1) - Z(0) + Z(0)Z(1) + X(1) - Z(0)X(1)\right)$$

eigenvalues: $\{0, 1, 2, 3\}$

$U =$ 

Task: Prepare eigenstates of H depending on qubit 2

qubit 2 in $|0\rangle$: Prepare ground state

qubit 2 in $|1\rangle$: Prepare highest eigenstate

```
H = H = 1.5-0.5*(Z(1)-Z(0)+Z(0)*Z(1)+X(1)-Z(0)*X(1))

U = Ry("x", 0) + Ry("xx", 0, control=2)
U+= CNOT(0,1) + Ry("y", 1) + Ry("yy", 1, control=2)

E0 = tq.ExpectationValue(H=H, U=U)
E1 = tq.ExpectationValue(H=H, U=tq.gates.X(2)+U)
objective = E0 - E1
result = tq.minimize(objective, initial_values="near_zero")
```

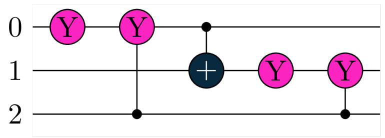
Two Strategies:

$$\min\ \left(\langle H\rangle_U - \langle H\rangle_{XU}\right)$$

$$\text{or:}\ \min\ \left(\langle P_{|000\rangle}\rangle_{UU_0^\dagger} + \langle P_{|001\rangle}\rangle_{XUU_1^\dagger}\right)$$

$$H = \frac{3}{2} - \frac{1}{2}\left(Z(1) - Z(0) + Z(0)Z(1) + X(1) - Z(0)X(1)\right)$$

eigenvalues: $\{0, 1, 2, 3\}$

$U =$ 

Task: Prepare eigenstates of H depending on qubit 2

qubit 2 in $|0\rangle$: Prepare ground state

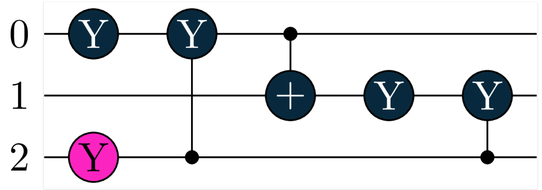
qubit 2 in $|1\rangle$: Prepare highest eigenstate

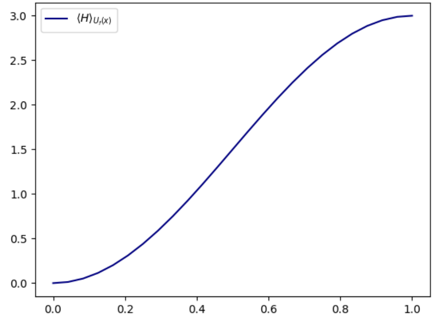
```
H = H = 1.5-0.5*(Z(1)-Z(0)+Z(0)*Z(1)+X(1)-Z(0)*X(1))

U = Ry("x", 0) + Ry("xx", 0, control=2)
U+= CNOT(0,1) + Ry("y", 1) + Ry("yy", 1, control=2)

E0 = tq.ExpectationValue(H=H, U=U)
E1 = tq.ExpectationValue(H=H, U=tq.gates.X(2)+U)
objective = E0 - E1
result = tq.minimize(objective, initial_values="near_zero")
```

Two Strategies:

$$\min \ \left(\langle H\rangle_U - \langle H\rangle_{XU}\right)$$

or: $$\min \ \left(\langle P_{|000\rangle}\rangle_{UU_0^\dagger} + \langle P_{|001\rangle}\rangle_{XUU_1^\dagger}\right)$$

$U_f =$ 




```
Uf = Ry(tq.Variable("control")*numpy.pi, 2)
Uf+= U.map_variables(result.variables)
f = tq.ExpectationValue(H=H, U=Uf)
f = tq.compile(f)
```

```
>>> f(0) = +1.00
    f(1) = +3.00
```

state_preparation_hamiltonian.py
state_preparation_fidelity.py

Measurement Optimization: Izmaylov Group

Implementation by Tzu-Ching “Thomson” Yen and Vladislav Verteletskyi

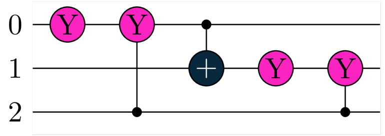
Yen, Verteletskyi, Izmaylov, JCTC, 2020

Verteletskyi, Yen, Izmaylov, JCP, 2020

$$H = \frac{3}{2} - \frac{1}{2}\left(Z(1) - Z(0) + Z(0)Z(1) + X(1) - Z(0)X(1)\right)$$

eigenvalues: $\{0, 1, 2, 3\}$

$U =$




```
E = tq.ExpectationValue(H=H, U=U)
print(E)
E_opt = tq.ExpectationValue(H=H, U=U, optimize_measurements=True)
print(E_opt)
```

```
>>> Objective with 1 unique expectation values
    total measurements = 6
    variables          = [a,b,c,d]
    types              = [not compiled]

    Objective with 2 unique expectation values
    total measurements = 2
    variables          = [a,b,c,d]
    types              = not compiled
```

faster when simulated

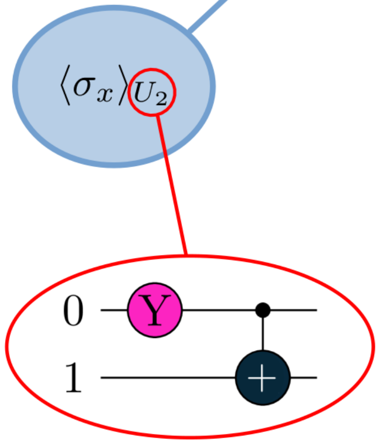
faster on quantum hardware

More Convenience with QTensor

$$\min \ (\langle H \rangle_U - \langle H \rangle_{XU})$$

```
E0 = tq.ExpectationValue(H=H, U=U)
E1 = tq.ExpectationValue(H=H, U=tq.gates.X(2)+U)
vector = tq.QTensor([E0, E1], shape=(2,))
weights = numpy.asarray([1.0,1.0])
objective = vector.dot(weights)
```

$$\begin{pmatrix} 0 & E_1 \\ E_2 & 0 \end{pmatrix} \begin{pmatrix} E_1 \\ 1 \end{pmatrix} \xrightarrow[\text{tq.simulate}]{a = 1.0, b = 0.5} - \begin{pmatrix} 0.841 \\ 0.738 \end{pmatrix}$$




```
import tequila as tq

U1 = tq.gates.Rx(angle="a", target=0)
U2 = tq.gates.Ry(angle="b", target=0)
U2 = tq.gates.CNOT(0,1)

H1 = tq.paulis.Y(0)
H2 = tq.paulis.X(0)+tq.paulis.Z(1)

E1 = tq.ExpectationValue(U1,H1)
E2 = tq.ExpectationValue(U2,H2)

M = tq.QTensor([0.0,E1,E2,0.0], shape=(2,2))
c = tq.QTensor([E1,1.0], shape=(2,))
b = M.dot(c)

variables={"a":1.0, "b":0.5}
result = tq.simulate(b, variables)
```

QOSF project by Gaurav Saxena (U Calgary)

see github/tequilahub/tequila-tutorials for more

$$\min \left(\langle P_{|000\rangle} \rangle_{UU_0^\dagger} + \langle P_{|001\rangle} \rangle_{XUU_1^\dagger} \right)$$

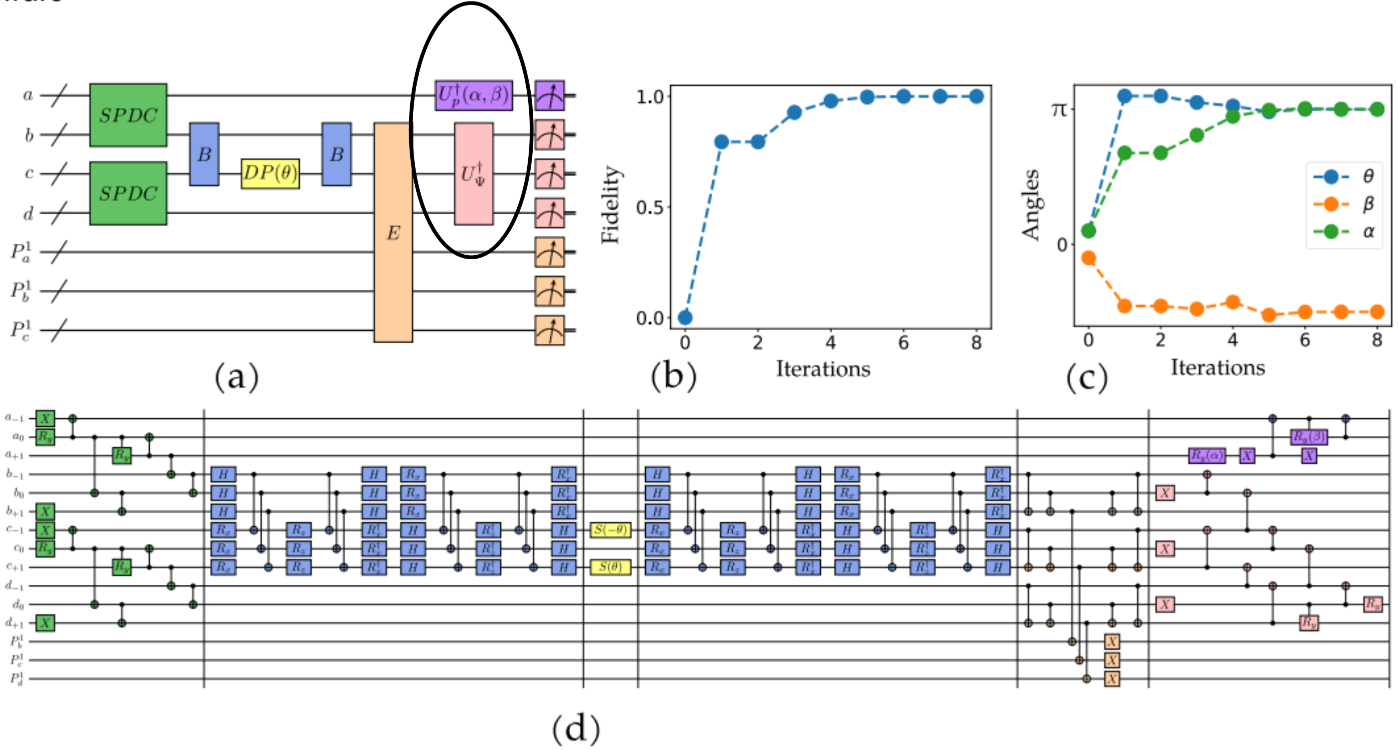
Application: Quantum Optics (same principle as in this example)

Quantum Science and Technology

PAPER

Quantum computer-aided design of quantum optics hardware

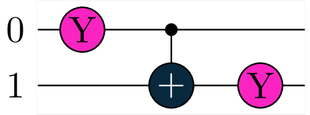
Jakob S Kottmann[1,2], Mario Krenn[1,2,3], Thi Ha Kyaw[1,2], Sumner Alperin-Lea[1]
and Alán Aspuru-Guzik[1,2,3,4,*]



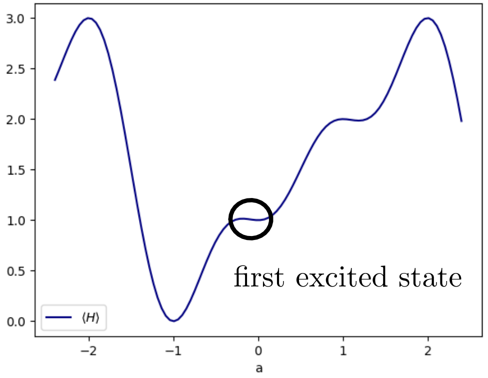
Example: Excited States

$$H = \frac{3}{2} - \frac{1}{2}\left(Z(1) - Z(0) + Z(0)Z(1) + X(1) - Z(0)X(1)\right)$$

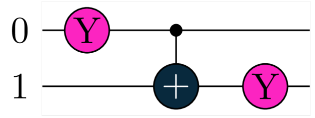
$U =$



Task: Prepare First Excited State



$$H = \frac{3}{2} - \frac{1}{2}\left(Z(1) - Z(0) + Z(0)Z(1) + X(1) - Z(0)X(1)\right)$$

$U =$ 

Task: Prepare First Excited State

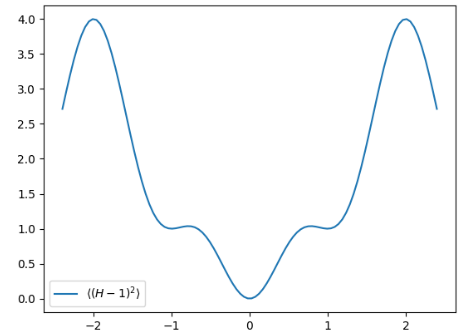
```
blue=tq.ExpectationValue(H=(H-1)**2, U=U)
```

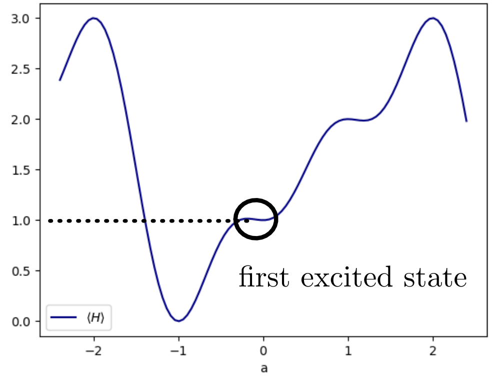
```
result=tq.minimize(blue)
```

$$\langle (H-1)^2 \rangle_U$$

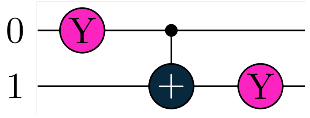
"folded spectrum"

McClean *et.al.* NJP 2016





$$H = \frac{3}{2} - \frac{1}{2}\left(Z(1) - Z(0) + Z(0)Z(1) + X(1) - Z(0)X(1)\right)$$

$U =$ 

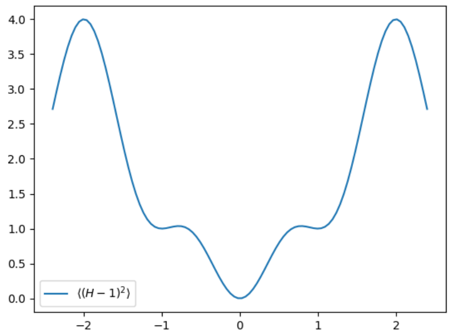
Task: Prepare First Excited State

```
blue=tq.ExpectationValue(H=(H-1)**2, U=U)
```

$$\langle (H-1)^2 \rangle_U$$

"folded spectrum"

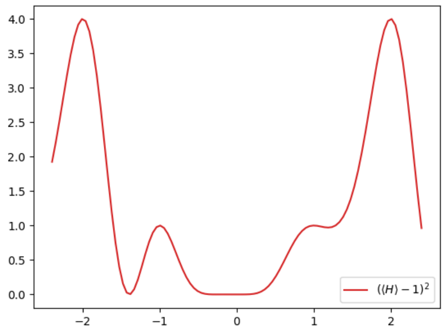
McClean *et.al.* NJP 2016




```
E = tq.ExpectationValue(H=H, U=U)
red = (E - 1.0)**2
```

$$(\langle H \rangle_U - 1)^2$$

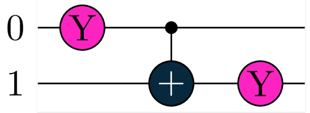
approximation




```
result=tq.minimize(red)
```

```
result=tq.minimize(blue)
```

$$H = \frac{3}{2} - \frac{1}{2}\left(Z(1) - Z(0) + Z(0)Z(1) + X(1) - Z(0)X(1)\right)$$

$U =$ 

Task: Prepare First Excited State

```
blue=tq.ExpectationValue(H=(H-1)**2, U=U)
```

$$\langle (H-1)^2 \rangle_U$$

"folded spectrum"

McClean *et.al.* NJP 2016

```
E = tq.ExpectationValue(H=H, U=U)
red = (E - 1.0)**2
```

$$(\langle H \rangle_U - 1)^2$$

approximation

```
U0 = U.map_variables({"a":-1.0})
U1 = U + U0.dagger()
E1 = tq.ExpectationValue(H=P, U=U1)
green = E - 10*E1
```

$$\langle H \rangle_U - 10 \langle P_{|00\rangle} \rangle_{U(a)U^\dagger(-1)}$$

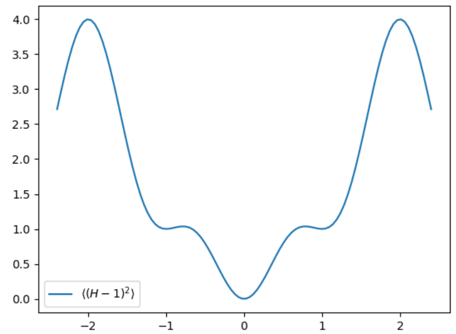
"VQD", "Orthogonality Constrained"

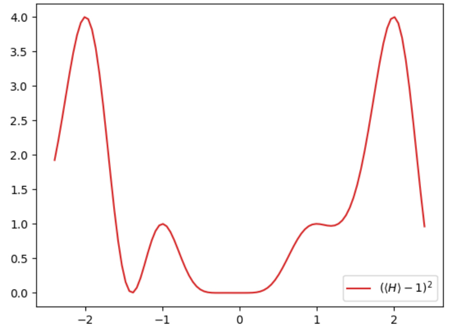
Lee *et.al*, JCTC, 2018

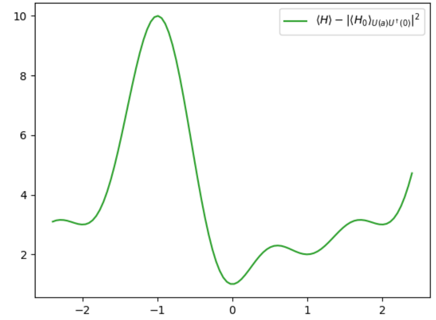
Higgot *et.al.* Quantum 2019

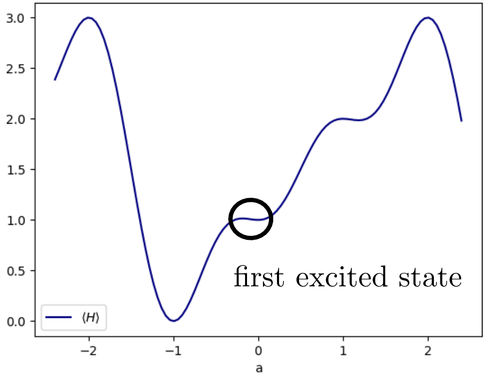
JSK, A. Anand, AAG, Chemical Science, 2021

explicit tequila examples









vqe_excited_state.py

$$\langle H \rangle_U - 10 \langle P_{|00\rangle} \rangle_{U(a)U^\dagger(-1)}$$

Application: Excited States in Chemistry

Chemical Science

ROYAL SOCIETY OF CHEMISTRY

EDGE ARTICLE

View Article Online
View Journal | View Issue

Check for updates

Cite this: *Chem. Sci.*, 2021, **12**, 3497

All publication charges for this article have been paid for by the Royal Society

A feasible approach for automatically differentiable unitary coupled-cluster on quantum computers†

Jakob S. Kottmann, *[ab] Abhinav Anand [a] and Alán Aspuru-Guzik *[abcd]

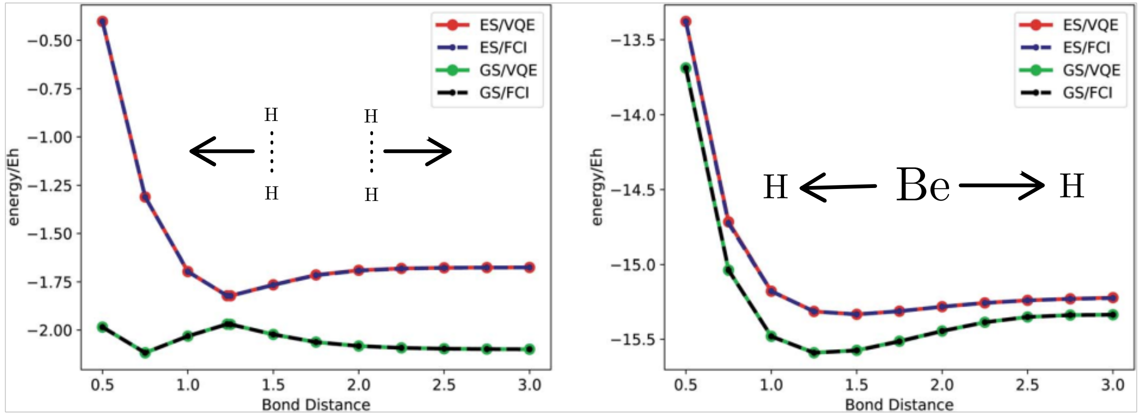


Fig. 6 Adapt-VQE for ground and excited states: adapt-VQE results using automatic differentiation for the screening and the optimization for ground and first excited state energies of H_4/STO-3G(4,8) (left) and BeH_2/STO-3G(6,14) (right). Except for the last point of the BeH_2 excited state, all points agree to millihartree accuracy with the corresponding exact solution (FCI) in the given basis set. See also Fig. 4. We included the special point at distance 1.23 Å with a square configuration.

$$\langle H \rangle_U - 10 \langle P_{|00\rangle} \rangle_{U(a)U^\dagger(-1)}$$

Application: Excited States in Chemistry

Chemical Science

ROYAL SOCIETY OF CHEMISTRY

EDGE ARTICLE

View Article Online
View Journal | View Issue

Check for updates

Cite this: *Chem. Sci.*, 2021, **12**, 3497

All publication charges for this article have been paid for by the Royal Society

A feasible approach for automatically differentiable unitary coupled-cluster on quantum computers†

Jakob S. Kottmann, *[ab] Abhinav Anand [a] and Alán Aspuru-Guzik *[abcd]

```
geometry = "H 0.0 0.0 0.0\nH 0.0 0.0 1.23\nH {R} 0.0 0.0\nH {R} 0.0 1.23"
mol = tq.Molecule(geometry=geometry.format(R=1.0), basis_set="sto-3g")
H = mol.make_hamiltonian()
U0 = mol.make_upccgsd_ansatz(name="UpCCGSD")
# ground state opt
E0 = tq.ExpectationValue(H=H, U=U0)
result0 = tq.minimize(E0)
U0_opt = U0.map_variables(result0.variables)
# start from CIS (see ChemicalScience SI)
U1 = tq.gates.X([0,1,2,3,4])
U1 += tq.gates.H(2)
U1 + tq.gates.CNOT(2,3)
U1 + tq.gates.CNOT(2,4)
U1 + tq.gates.CNOT(2,5)
# excited state ansatz (CIS + UpCCGSD)
U1 = U1 + mol.make_upccgsd_ansatz(name="UpCCGSD", include_reference=False)
E1 = tq.ExpectationValue(H=H, U=U1)
P0 = tq.paulis.Qp(U1.qubits) #same as |0><0|
S = tq.ExpectationValue(H=P0, U=U1+U0_opt.dagger())
f_ex = E1 - result0.energy*S
result1 = tq.minimize(f_ex)
```

vqe_ex_chem.py

```
vqe ground state : -1.9771
vqe excited state: -1.7326
walltime: 11s
```

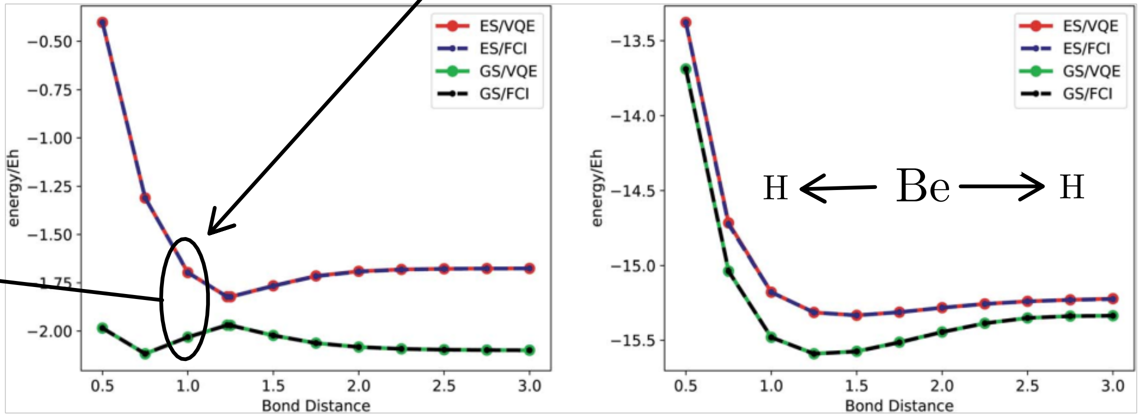


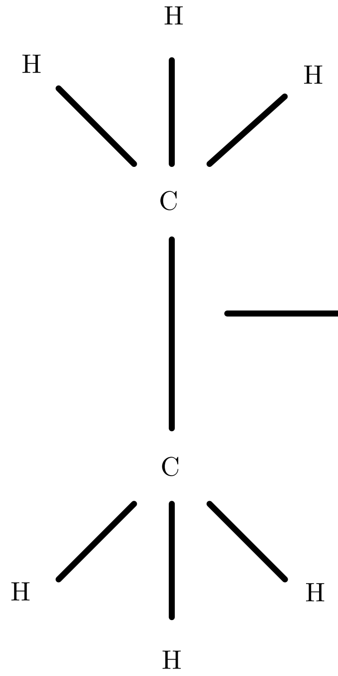
Fig. 6 Adapt-VQE for ground and excited states: adapt-VQE results using automatic differentiation for the screening and the optimization for ground and first excited state energies of H_4/STO-3G(4,8) (left) and BeH_2/STO-3G(6,14) (right). Except for the last point of the BeH_2 excited state, all points agree to millihartree accuracy with the corresponding exact solution (FCI) in the given basis set. See also Fig. 4. We included the special point at distance 1.23 Å with a square configuration.

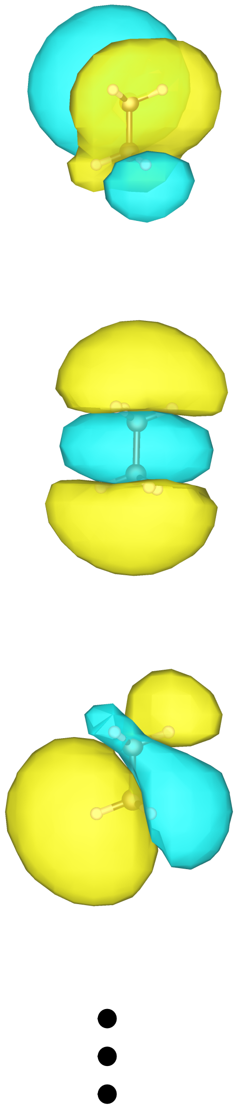
in paper: Adaptive VQE

see "Adaptive Solver" Tutorial

github.com/tequilahub/tequila-tutorials

Overview: QuantumChemistry





default: System adapted orbitals

1 spatial orbital for each electron

need more orbitals?

```
tq.Molecule(geometry="c2h6.xyz", n_qubits=28
```

"basis sets" also possible

```
tq.Molecule(geometry="c2h6.xyz", basis_set="sto-3g"
```

map to qubits

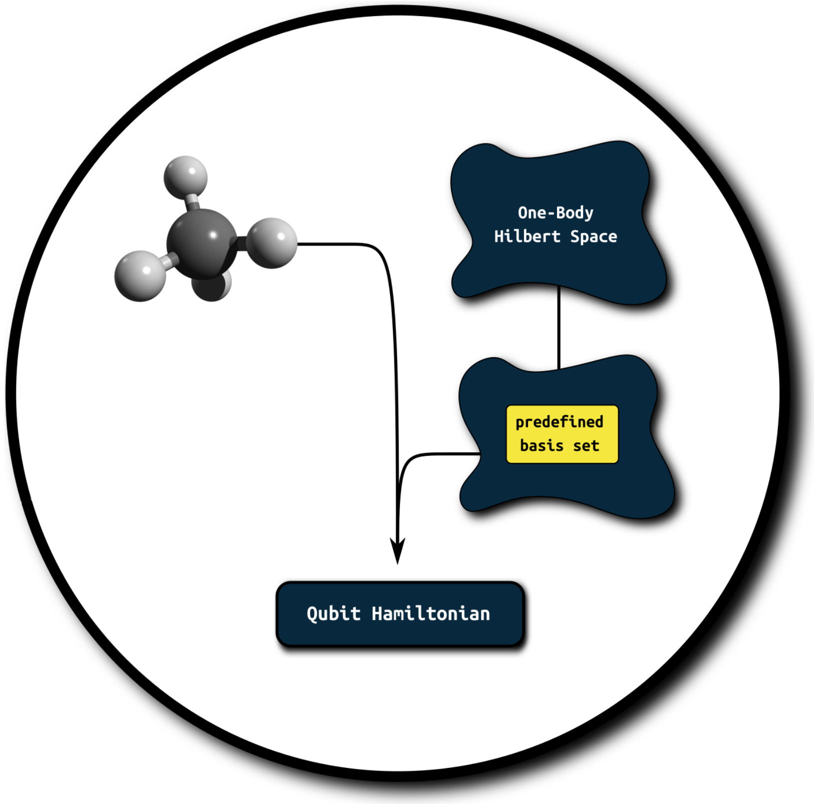
example

```
import tequila as tq

mol = tq.Molecule(geometry="c2h6.xyz")
H_HCB = mol.make_hardcore_boson_hamiltonian()
U_HCB = mol.make_ansatz("HCB-SPA")
E = tq.ExpectationValue(H=H_HCB, U=U_HCB)

result = tq.minimize(E)
```

HCB: Hard-Core-Boson approximation

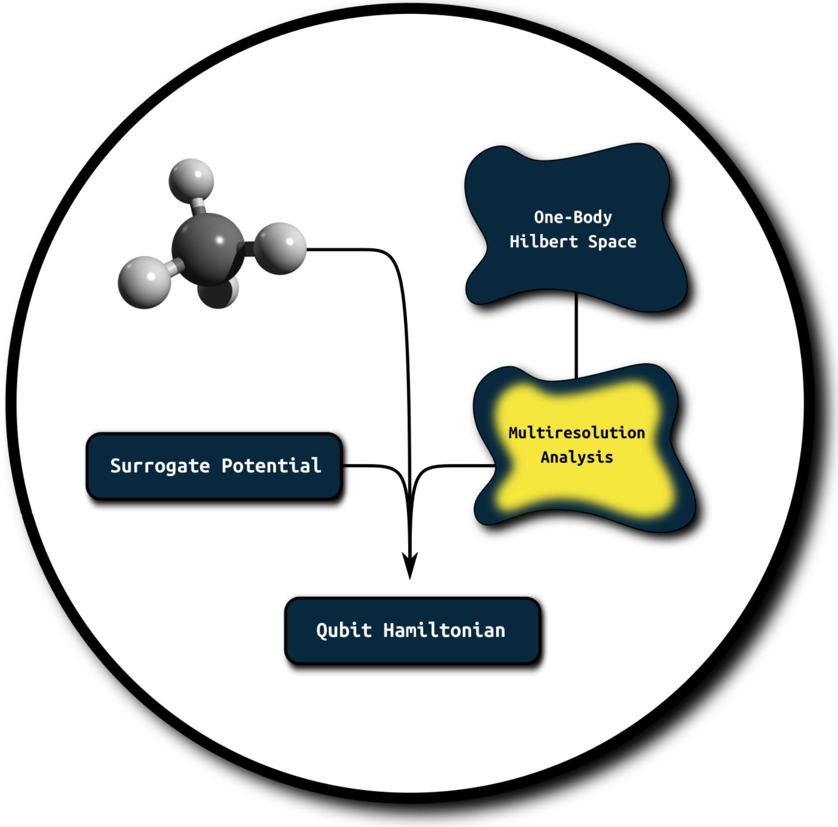
requires only half the qubits (here 14 instead of 28)

c2h6.py




```
mol = tq.Molecule(geometry="ch4.xyz", basis_set="sto-3g")

H = mol.make_hamiltonian()
U = mol.make_ansatz(name="UpCCD")
E = tq.ExpectationValue(H=H, U=U)

result = tq.minimize(E)
exact =  tq.compute_energy("fci")
```

```
System Adapted:

8 orbitals
VQE/MRA-PNO : -40.2761
FCI/MRA-PNO : -40.2926

Basis Sets:

9 orbitals
VQE/STO-3G   : -39.7580
FCI/STO-3G   : -39.8060

17 orbitals
FCI/6-31G    : -40.3013
```




```
mol = tq.Molecule(geometry="ch4.xyz")

H = mol.make_hamiltonian()
U = mol.make_ansatz(name="UpCCD")
E = tq.ExpectationValue(H=H, U=U)

result = tq.minimize(E)
exact =  tq.compute_energy("fci")
```

backends: PySCF or Psi4

backend: MADNESS

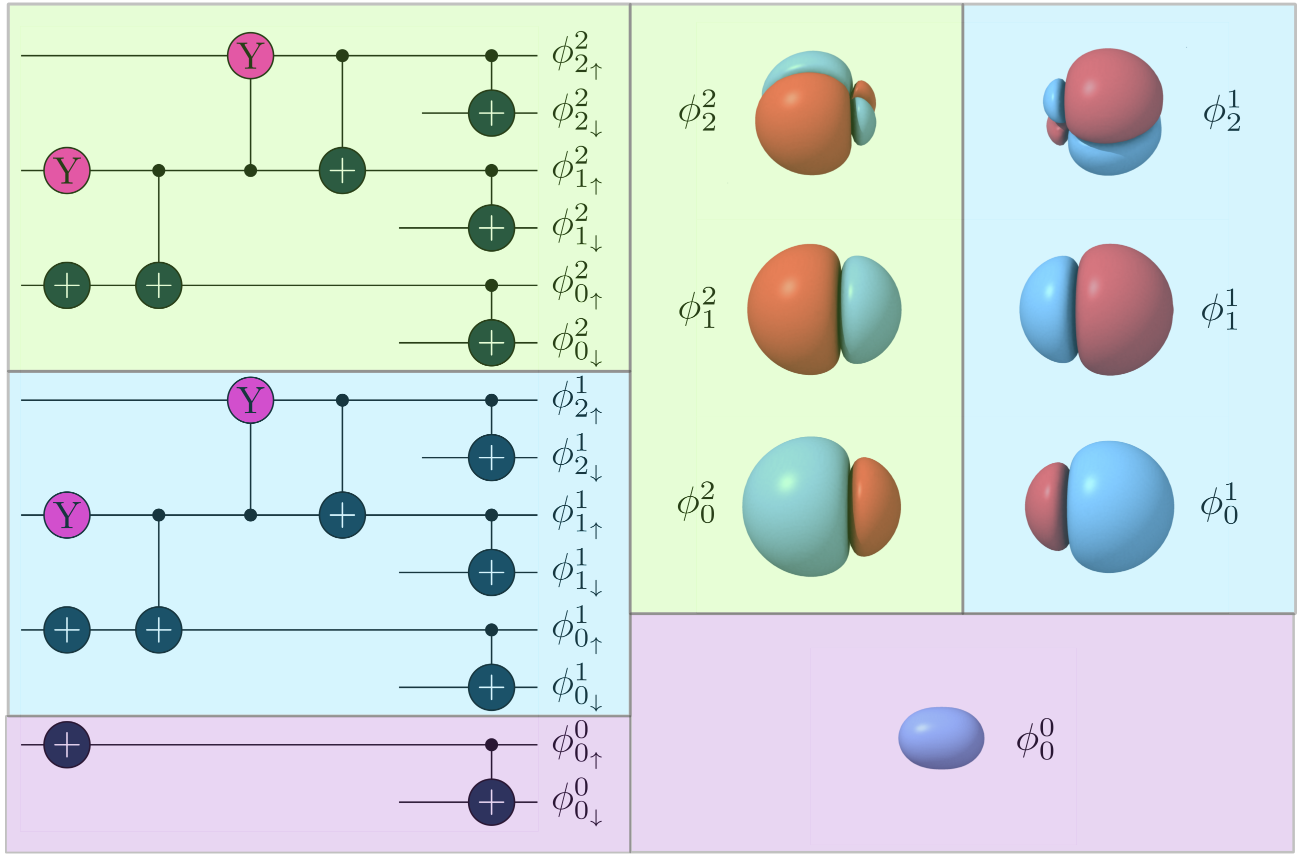
ch4.py

Application: Dequantized VQE via Separable Pair Approximations (SPA)

Optimized Low-Depth Quantum Circuits for Molecular Electronic Structure using a Separable Pair Approximation

Jakob S. Kottmann[1,2,*] and Alán Aspuru-Guzik[1,2,3,4,†]

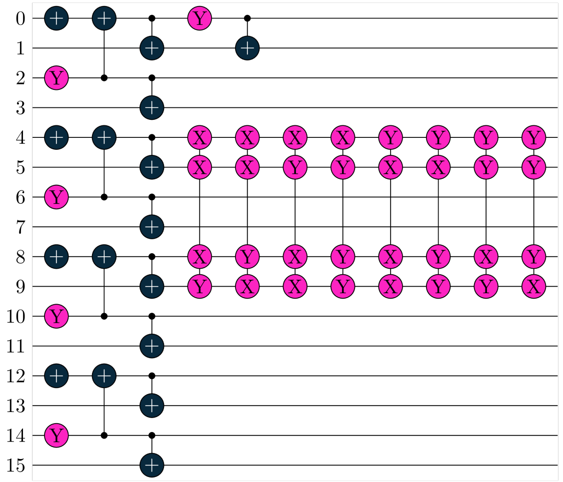
Molecule(N_e,N_q)	N_{param}	N_{cnot}	Depth
H_2(2,4)	1	3	3
LiH(2,10)	4	15	18
BeH_2(4,8)	2	6	3
BeH_2(6,14)	4	15	7
BH_3(6,12)	3	9	3
N_2(6,12)	3	9	3
C_2H_4(12,24)	6	18	3
H_2O_2(14,28)	7	21	3
C_2H_6(14,28)	7	21	3
C_2H_6(2,12)	5	19	23
C_2H_6(14,84)	35	133	23

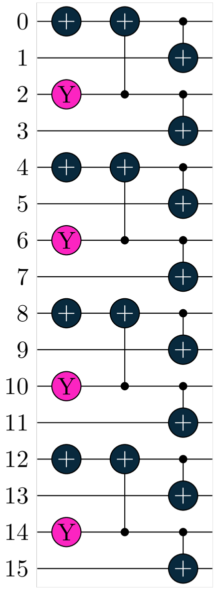


Optimized Low-Depth Quantum Circuits for Molecular Electronic Structure using a Separable Pair Approximation

Jakob S. Kottmann[1, 2, *] and Alán Aspuru-Guzik[1, 2, 3, 4, †]

```
U = mol.make_ansatz("SPA")
U += tq.gates.QubitExcitation("a", [2,3,4,5])
U += tq.gates.Ry(angle="b",target=0)
U+= tq.gates.CNOT(0,1)
```

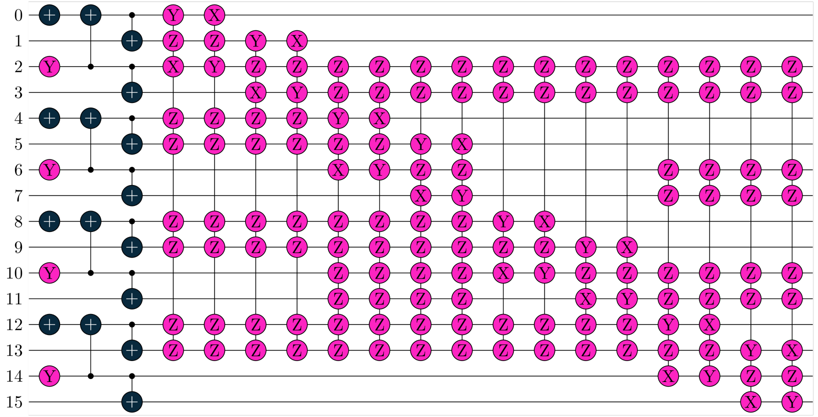




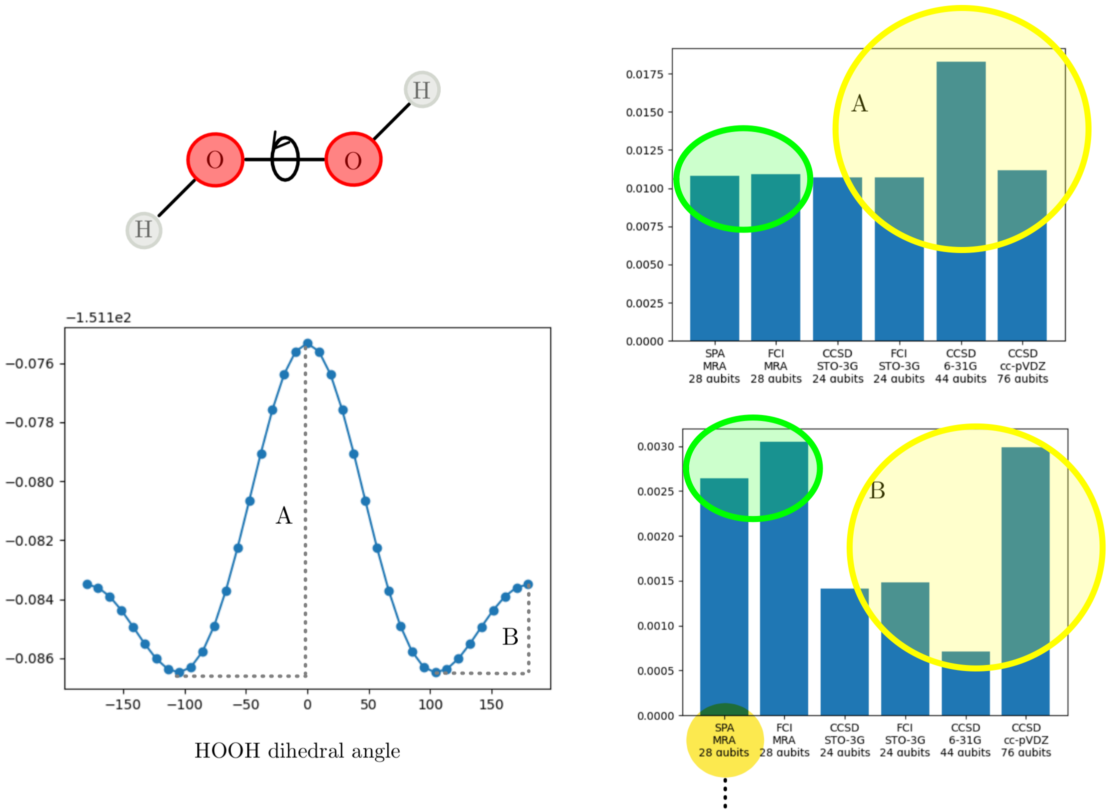
make your own ...

```
mol.make_ansatz("SPA")
```

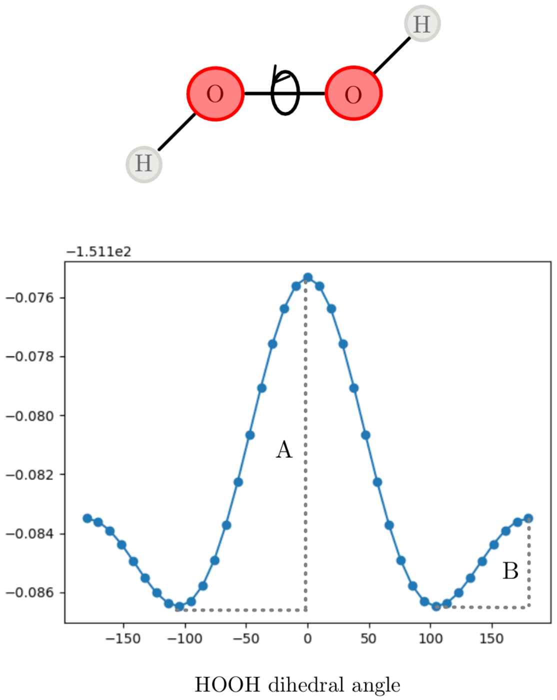
```
mol.make_ansatz("SPA+S")
```

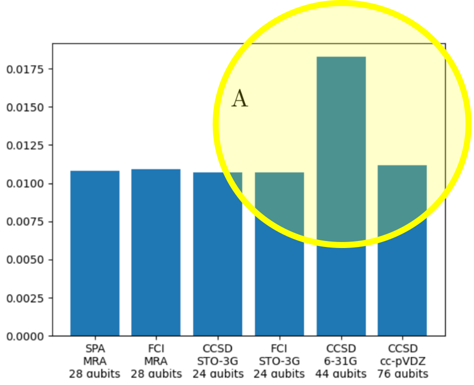


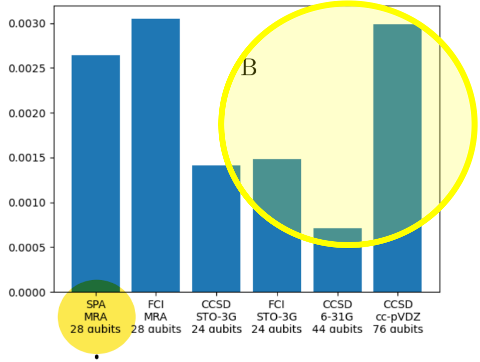
ch4_circuits.py




```
import tequila as tq

mol = tq.Molecule(geometry="h2o2.xyz", n_pno=7, pno={"maxrank":1})
H = mol.make_hardcore_boson_hamiltonian()
U = mol.make_upccgsd_ansatz(name="HCB-SPA")
E = tq.ExpectationValue(H=H,U=U)

result=tq.minimize(E)
energy=result.energy
```



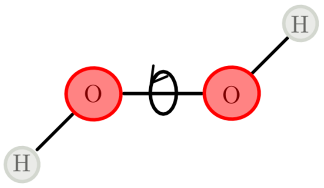


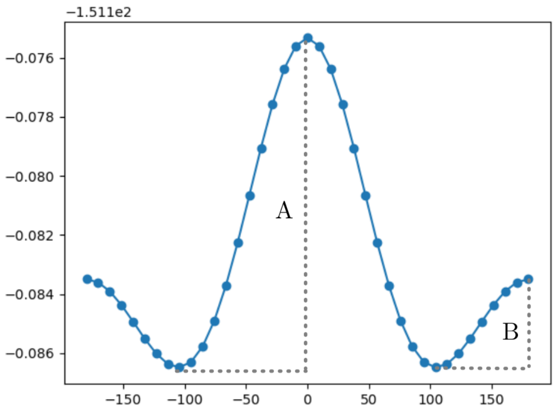



```
import tequila as tq

mol = tq.Molecule(geometry="h2o2.xyz", n_pno=7, pno={"maxrank":1})
H = mol.make_hardcore_boson_hamiltonian()
U = mol.make_upccgsd_ansatz(name="HCB-SPA")
E = tq.ExpectationValue(H=H,U=U)

result=tq.minimize(E)
energy=result.energy
```

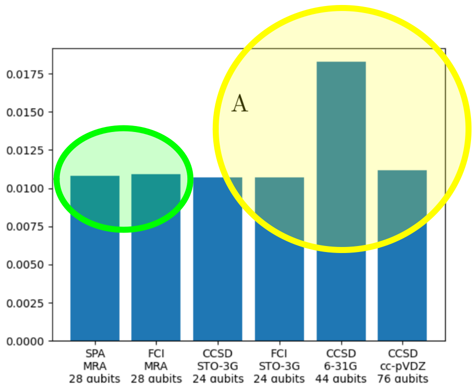
In this example:

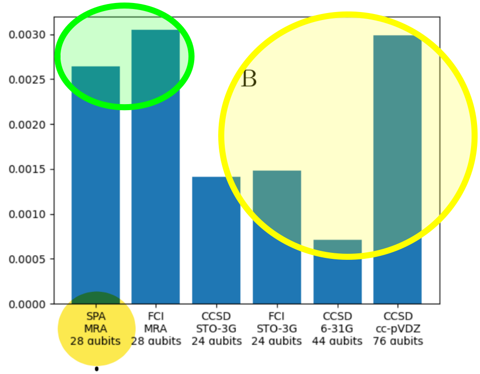
larger basis set is not always better





HOOH dihedral angle






```
import tequila as tq

mol = tq.Molecule(geometry="h2o2.xyz", n_pno=7, pno={"maxrank":1})
H = mol.make_hardcore_boson_hamiltonian()
U = mol.make_upccgsd_ansatz(name="HCB-SPA")
E = tq.ExpectationValue(H=H,U=U)

result=tq.minimize(E)
energy=result.energy
```

In this example:

larger basis set is not always better

SPA is a good approximation

MRA-PNOs are accurate and compact

”more accuracy with less qubits”

THE JOURNAL OF PHYSICAL CHEMISTRY LETTERS
A JOURNAL OF THE AMERICAN CHEMICAL SOCIETY
pubs.acs.org/JPCL Letter

Reducing Qubit Requirements while Maintaining Numerical Precision for the Variational Quantum Eigensolver: A Basis-Set-Free Approach

Jakob S. Kottmann,* Philipp Schleich, Teresa Tamayo-Mendoza, and Alán Aspuru-Guzik*

high level blog entry:

https://aspuru.substack.com/p/bits-are-cheap-and-qubits-expensive

Alán
Aspuru-Guzik
UofT

Philipp
Schleich
UofT/CS

Abhinav
Anand
UofT/Chem

Sumner
Alperin-Lea
UofT/Chem

Alba
Cervera-Lierta
Barcelona Supercomputing Center

UNIVERSITY OF TORONTO

the matter lab

Phillip
Jensen
UofT/Chem

Thi Ha
Kyaw
UofT/CS

Teresa
Tamayo-Mendoza
Harvard

Mario
Krenn
MPI Erlangen

Zi-Jian
Zhang
UofT/CS

qosf

Initial Team: Sumner Alperin-Lea, Alba Cervera-Lierta, Teresa Tamayo-Medoza, Cyrille Lavigne

AAG-Group (UofT): Philipp Schleich, Abhinav Anand, Matthias Degroote, Skylar Chaney, Maha Kesibi, Naomi Grace Curnow, Arkaprava Choudhurry

Izmaylov-Group (UofT): Tzu-Ching "Thomson" Yen, Vladyslav Verteletskyi, Zachary Bansingh

QOSF: Brandon Solo, Giorgios Tsilimigkounaikis,Claudia Zendeja-Morales, Tanya Garg

Github: Alejandro de la Serna, Leo Becker, David Wierichs, Arianne van den Griend .... You?

DS3Lab (ETH): Maurice Weber

tequila

github/tequilahub