group-theory-square-molecule

April 11, 2024

[ ]:
```python
from sympy.physics.quantum.state import Bra, Ket
from sympy.physics.quantum.dagger import Dagger
from sympy.physics.quantum import qapply, Bra, Ket
from sympy.matrices import Matrix
from sympy import expand, symbols, exp, I, re
import numpy as np
```

0.1 (1) Definition of states

[ ]:
```python
sa, sb, sc, sd = Ket("SA"), Ket("SB"), Ket("SC"), Ket("SD")
pax, pbx, pcx, pdx = Ket("PAx"), Ket("PBx"), Ket("PCx"), Ket("PDx")
pay, pby, pcy, pdy = Ket("PAy"), Ket("PBy"), Ket("PCy"), Ket("PDy")
paz, pbz, pcz, pdz = Ket("PAz"), Ket("PBz"), Ket("PCz"), Ket("PDz")
```

[ ]:
```python
Phi = Matrix(
    [
        # Gamma(0)
        0.5 * (sa + sd + sc + sb),
        0.5 * (pax + pdy + pcx + pby),
        0.5 * (pay + pdx + pcy + pbx),
        0.5 * (paz + pdz + pcz + pbz),
        # Gamma(1)
        0.5 * (sa - I * sd - sc + I * sb),
        0.5 * (pax - I * pdy - pcx + I * pby),
        0.5 * (pay - I * pdx - pcy + I * pbx),
        0.5 * (paz - I * pdz - pcz + I * pbz),
        # Gamma(2)
        0.5 * (sa - sd + sc - sb),
        0.5 * (pax - pdy + pcx - pby),
        0.5 * (pay - pdx + pcy - pbx),
        0.5 * (paz - pdz + pcz - pbz),
        # Gamma(3)
        0.5 * (sa + I * sd - sc - I * sb),
        0.5 * (pax + I * pdy - pcx - I * pby),
        0.5 * (pay + I * pdx - pcy - I * pbx),
        0.5 * (paz + I * pdz - pcz - I * pbz),
    ]
```

```
).T

Phi = expand(Phi)
```

0.2 (2) Hamiltonian

NOTE: we don't sandwhich bras and kets around "true" hamiltonia operator here as we will substitute them with predefined symbols (or precalculated elements such as E_s, E_p, $ss\sigma$)

[ ]:
```
# inner product of bras and kets
h = Phi.applyfunc(lambda i: Dagger(i)).T * Phi
# sympy syntax to apply quantum operation
h = h.applyfunc(lambda i: qapply(expand(i)).doit())
```

[ ]:
```
H = h
Es = symbols("Es", real=True)
Ep = symbols("Ep", real=True)
sss = symbols(r"ss\sigma", real=True)
sps = symbols(r"sp\sigma", real=True)
pps = symbols(r"ps\sigma", real=True)
ppp = symbols(r"pp\pi", real=True)

# atomic orbital basis
orbitals = [sa, sb, sc, sd, pax, pay, paz, pbx, pby, pbz, pcx, pcy, pcz, pdx,␣
 ↪pdy, pdz]

circle = "ABCD"

order = len(circle)

# repalce predefined inner products of bras and kets
for oi in orbitals:
    for oj in orbitals:
        if oi == oj and oi in [sa, sb, sc, sd]:
            H = H.subs(oi.dual * oj, Es)
            continue
        elif oi == oj and oi in [
            pax,
            pay,
            paz,
            pbx,
            pby,
            pbz,
            pcx,
            pcy,
            pcz,
            pdx,
```

```python
            pdy,
            pdz,
        ]:
            H = H.subs(oi.dual * oj, Ep)
            continue

        stri, strj = str(oi), str(oj)

        i = circle.index(stri[2])
        j = circle.index(strj[2])

        dist = min((i - j) % order, (j - i) % order)

        if dist == 2:
            H = H.subs(oi.dual * oj, 0)
            continue
        if dist == 0 and oi != oj:
            H = H.subs(oi.dual * oj, 0)
            continue

        if stri[1] == "S" and strj[1] == "S":
            H = H.subs(oi.dual * oj, sss)
            continue
        orbset = set([oi, oj])
        if (
            orbset == set([sa, pbx])
            or orbset == set([sb, pax])
            or orbset == set([sc, pdx])
            or orbset == set([sd, pcx])
            or orbset == set([sa, pdy])
            or orbset == set([sb, pcy])
            or orbset == set([sc, pby])
            or orbset == set([sd, pay])
        ):
            H = H.subs(oi.dual * oj, sps)
        elif (
            orbset == set([pax, pbx])
            or orbset == set([pby, pcy])
            or orbset == set([pcx, pdx])
            or orbset == set([pdy, pay])
        ):
            H = H.subs(oi.dual * oj, pps)
        elif (
            orbset == set([pax, pdx])
            or orbset == set([pbx, pcx])
            or orbset == set([pcy, pdy])
            or orbset == set([pay, pby])
```

```
        ):
            H = H.subs(oi.dual * oj, ppp)
        else:
            H = H.subs(oi.dual * oj, 0)
```

[ ]:
```
H
```

[ ]:

$$\left[\begin{matrix}1.0 Es + 2.0 ss\sigma & 1.0 sp\sigma & 1.0 sp\sigma & 0 & 0 & 0 & 0\\1.0 sp\sigma & 1.0 Ep & 1.0 pp\pi + 1.0 ps\sigma & 0 & 0 & 0 & 0\\1.0 sp\sigma & 1.0 pp\pi + 1.0 ps\sigma & 1.0 Ep & 0 & 0 & 0 & 0\\0 & 0 & 0 & 1.0 Ep & 0 & 0 & 0\\0 & 0 & 0 & 0 & 1.0 Es & -1.0 i sp\sigma & 1.0 i sp\sigma\\0 & 0 & 0 & 0 & 1.0 i sp\sigma & 1.0 Ep & -1.0 i pp\pi + 1.0 i ps\sigma\\0 & 0 & 0 & 0 & -1.0 i sp\sigma & 1.0 i pp\pi - 1.0 i ps\sigma & 1.0 Ep\\0 & 0 & 0 & 0 & 0 & 0 & 0\\0 & 0 & 0 & 0 & 0 & 0 & 0\\0 & 0 & 0 & 0 & 0 & 0 & 0\\0 & 0 & 0 & 0 & 0 & 0 & 0\\0 & 0 & 0 & 0 & 0 & 0 & 0\\0 & 0 & 0 & 0 & 0 & 0 & 0\\0 & 0 & 0 & 0 & 0 & 0 & 0\\0 & 0 & 0 & 0 & 0 & 0 & 0\\0 & 0 & 0 & 0 & 0 & 0 & 0\end{matrix}\right.$$

0.3 (3) Eigenvalues and eigenfunctions

[ ]:
```
H.eigenvals()
```

[ ]:
```
{0.5*Ep + 0.5*Es + 0.5*pp\pi + 0.5*ps\sigma + 1.0*ss\sigma -
1.4142135623731*sqrt(0.125*Ep**2 - 0.25*Ep*Es + 0.25*Ep*pp\pi + 0.25*Ep*ps\sigma
- 0.5*Ep*ss\sigma + 0.125*Es**2 - 0.25*Es*pp\pi - 0.25*Es*ps\sigma +
0.5*Es*ss\sigma + 0.125*pp\pi**2 + 0.25*pp\pi*ps\sigma - 0.5*pp\pi*ss\sigma +
0.125*ps\sigma**2 - 0.5*ps\sigma*ss\sigma + 1.0*sp\sigma**2 + 0.5*ss\sigma**2):
1,
 0.5*Ep + 0.5*Es + 0.5*pp\pi + 0.5*ps\sigma + 1.0*ss\sigma +
1.4142135623731*sqrt(0.125*Ep**2 - 0.25*Ep*Es + 0.25*Ep*pp\pi + 0.25*Ep*ps\sigma
- 0.5*Ep*ss\sigma + 0.125*Es**2 - 0.25*Es*pp\pi - 0.25*Es*ps\sigma +
0.5*Es*ss\sigma + 0.125*pp\pi**2 + 0.25*pp\pi*ps\sigma - 0.5*pp\pi*ss\sigma +
0.125*ps\sigma**2 - 0.5*ps\sigma*ss\sigma + 1.0*sp\sigma**2 + 0.5*ss\sigma**2):
1,
 1.0*Ep - 1.0*pp\pi - 1.0*ps\sigma: 1,
 1.0*Ep: 4,
 0.666666666666667*Ep + 0.333333333333333*Es - 0.111111111111111*(-3.0*Ep**2 -
6.0*Ep*Es + 3.0*pp\pi**2 - 6.0*pp\pi*ps\sigma + 3.0*ps\sigma**2 +
6.0*sp\sigma**2 + 4.0*(-Ep - 0.5*Es)**2)/(-0.5*Ep**2*Es + Ep*sp\sigma**2 +
0.5*Es*pp\pi**2 - Es*pp\pi*ps\sigma + 0.5*Es*ps\sigma**2 -
0.0185185185185185*(-18.0*Ep - 9.0*Es)*(1.0*Ep**2 + 2.0*Ep*Es - 1.0*pp\pi**2 +
```

```
2.0*pp\pi*ps\sigma - 1.0*ps\sigma**2 - 2.0*sp\sigma**2) + 0.296296296296296*(-Ep
- 0.5*Es)**3 + sqrt(-0.296296296296296*(-0.5*Ep**2 - Ep*Es + 0.5*pp\pi**2 -
pp\pi*ps\sigma + 0.5*ps\sigma**2 + sp\sigma**2 + 0.666666666666667*(-Ep -
0.5*Es)**2)**3 + (-0.5*Ep**2*Es + Ep*sp\sigma**2 + 0.5*Es*pp\pi**2 -
Es*pp\pi*ps\sigma + 0.5*Es*ps\sigma**2 - 0.0185185185185185*(-18.0*Ep -
9.0*Es)*(1.0*Ep**2 + 2.0*Ep*Es - 1.0*pp\pi**2 + 2.0*pp\pi*ps\sigma -
1.0*ps\sigma**2 - 2.0*sp\sigma**2) + 0.296296296296296*(-Ep -
0.5*Es)**3)**2))**(1/3) - 1.0*(-0.5*Ep**2*Es + Ep*sp\sigma**2 + 0.5*Es*pp\pi**2
- Es*pp\pi*ps\sigma + 0.5*Es*ps\sigma**2 - 0.0185185185185185*(-18.0*Ep -
9.0*Es)*(1.0*Ep**2 + 2.0*Ep*Es - 1.0*pp\pi**2 + 2.0*pp\pi*ps\sigma -
1.0*ps\sigma**2 - 2.0*sp\sigma**2) + 0.296296296296296*(-Ep - 0.5*Es)**3 +
sqrt(-0.296296296296296*(-0.5*Ep**2 - Ep*Es + 0.5*pp\pi**2 - pp\pi*ps\sigma +
0.5*ps\sigma**2 + sp\sigma**2 + 0.666666666666667*(-Ep - 0.5*Es)**2)**3 +
(-0.5*Ep**2*Es + Ep*sp\sigma**2 + 0.5*Es*pp\pi**2 - Es*pp\pi*ps\sigma +
0.5*Es*ps\sigma**2 - 0.0185185185185185*(-18.0*Ep - 9.0*Es)*(1.0*Ep**2 +
2.0*Ep*Es - 1.0*pp\pi**2 + 2.0*pp\pi*ps\sigma - 1.0*ps\sigma**2 -
2.0*sp\sigma**2) + 0.296296296296296*(-Ep - 0.5*Es)**3)**2))**(1/3): 2,
 0.666666666666667*Ep + 0.333333333333333*Es - 0.111111111111111*(-3.0*Ep**2 -
6.0*Ep*Es + 3.0*pp\pi**2 - 6.0*pp\pi*ps\sigma + 3.0*ps\sigma**2 +
6.0*sp\sigma**2 + 4.0*(-Ep - 0.5*Es)**2)/((-1/2 + sqrt(3)*I/2)*(-0.5*Ep**2*Es +
Ep*sp\sigma**2 + 0.5*Es*pp\pi**2 - Es*pp\pi*ps\sigma + 0.5*Es*ps\sigma**2 -
0.0185185185185185*(-18.0*Ep - 9.0*Es)*(1.0*Ep**2 + 2.0*Ep*Es - 1.0*pp\pi**2 +
2.0*pp\pi*ps\sigma - 1.0*ps\sigma**2 - 2.0*sp\sigma**2) + 0.296296296296296*(-Ep
- 0.5*Es)**3 + sqrt(-0.296296296296296*(-0.5*Ep**2 - Ep*Es + 0.5*pp\pi**2 -
pp\pi*ps\sigma + 0.5*ps\sigma**2 + sp\sigma**2 + 0.666666666666667*(-Ep -
0.5*Es)**2)**3 + (-0.5*Ep**2*Es + Ep*sp\sigma**2 + 0.5*Es*pp\pi**2 -
Es*pp\pi*ps\sigma + 0.5*Es*ps\sigma**2 - 0.0185185185185185*(-18.0*Ep -
9.0*Es)*(1.0*Ep**2 + 2.0*Ep*Es - 1.0*pp\pi**2 + 2.0*pp\pi*ps\sigma -
1.0*ps\sigma**2 - 2.0*sp\sigma**2) + 0.296296296296296*(-Ep -
0.5*Es)**3)**2))**(1/3)) - 1.0*(-1/2 + sqrt(3)*I/2)*(-0.5*Ep**2*Es +
Ep*sp\sigma**2 + 0.5*Es*pp\pi**2 - Es*pp\pi*ps\sigma + 0.5*Es*ps\sigma**2 -
0.0185185185185185*(-18.0*Ep - 9.0*Es)*(1.0*Ep**2 + 2.0*Ep*Es - 1.0*pp\pi**2 +
2.0*pp\pi*ps\sigma - 1.0*ps\sigma**2 - 2.0*sp\sigma**2) + 0.296296296296296*(-Ep
- 0.5*Es)**3 + sqrt(-0.296296296296296*(-0.5*Ep**2 - Ep*Es + 0.5*pp\pi**2 -
pp\pi*ps\sigma + 0.5*ps\sigma**2 + sp\sigma**2 + 0.666666666666667*(-Ep -
0.5*Es)**2)**3 + (-0.5*Ep**2*Es + Ep*sp\sigma**2 + 0.5*Es*pp\pi**2 -
Es*pp\pi*ps\sigma + 0.5*Es*ps\sigma**2 - 0.0185185185185185*(-18.0*Ep -
9.0*Es)*(1.0*Ep**2 + 2.0*Ep*Es - 1.0*pp\pi**2 + 2.0*pp\pi*ps\sigma -
1.0*ps\sigma**2 - 2.0*sp\sigma**2) + 0.296296296296296*(-Ep -
0.5*Es)**3)**2))**(1/3): 2,
 0.666666666666667*Ep + 0.333333333333333*Es - 0.111111111111111*(-3.0*Ep**2 -
6.0*Ep*Es + 3.0*pp\pi**2 - 6.0*pp\pi*ps\sigma + 3.0*ps\sigma**2 +
6.0*sp\sigma**2 + 4.0*(-Ep - 0.5*Es)**2)/((-1/2 - sqrt(3)*I/2)*(-0.5*Ep**2*Es +
Ep*sp\sigma**2 + 0.5*Es*pp\pi**2 - Es*pp\pi*ps\sigma + 0.5*Es*ps\sigma**2 -
0.0185185185185185*(-18.0*Ep - 9.0*Es)*(1.0*Ep**2 + 2.0*Ep*Es - 1.0*pp\pi**2 +
2.0*pp\pi*ps\sigma - 1.0*ps\sigma**2 - 2.0*sp\sigma**2) + 0.296296296296296*(-Ep
- 0.5*Es)**3 + sqrt(-0.296296296296296*(-0.5*Ep**2 - Ep*Es + 0.5*pp\pi**2 -
```

```
pp\pi*ps\sigma + 0.5*ps\sigma**2 + sp\sigma**2 + 0.666666666666667*(-Ep -
0.5*Es)**2)**3 + (-0.5*Ep**2*Es + Ep*sp\sigma**2 + 0.5*Es*pp\pi**2 -
Es*pp\pi*ps\sigma + 0.5*Es*ps\sigma**2 - 0.0185185185185185*(-18.0*Ep -
9.0*Es)*(1.0*Ep**2 + 2.0*Ep*Es - 1.0*pp\pi**2 + 2.0*pp\pi*ps\sigma -
1.0*ps\sigma**2 - 2.0*sp\sigma**2) + 0.296296296296296*(-Ep -
0.5*Es)**3)**2))**(1/3)) - 1.0*(-1/2 - sqrt(3)*I/2)*(-0.5*Ep**2*Es +
Ep*sp\sigma**2 + 0.5*Es*pp\pi**2 - Es*pp\pi*ps\sigma + 0.5*Es*ps\sigma**2 -
0.0185185185185185*(-18.0*Ep - 9.0*Es)*(1.0*Ep**2 + 2.0*Ep*Es - 1.0*pp\pi**2 +
2.0*pp\pi*ps\sigma - 1.0*ps\sigma**2 - 2.0*sp\sigma**2) + 0.296296296296296*(-Ep
- 0.5*Es)**3 + sqrt(-0.296296296296296*(-0.5*Ep**2 - Ep*Es + 0.5*pp\pi**2 -
pp\pi*ps\sigma + 0.5*ps\sigma**2 + sp\sigma**2 + 0.666666666666667*(-Ep -
0.5*Es)**2)**3 + (-0.5*Ep**2*Es + Ep*sp\sigma**2 + 0.5*Es*pp\pi**2 -
Es*pp\pi*ps\sigma + 0.5*Es*ps\sigma**2 - 0.0185185185185185*(-18.0*Ep -
9.0*Es)*(1.0*Ep**2 + 2.0*Ep*Es - 1.0*pp\pi**2 + 2.0*pp\pi*ps\sigma -
1.0*ps\sigma**2 - 2.0*sp\sigma**2) + 0.296296296296296*(-Ep -
0.5*Es)**3)**2))**(1/3): 2,
 0.5*Ep + 0.5*Es - 0.5*pp\pi - 0.5*ps\sigma - 1.0*ss\sigma -
1.4142135623731*sqrt(0.125*Ep**2 - 0.25*Ep*Es - 0.25*Ep*pp\pi - 0.25*Ep*ps\sigma
+ 0.5*Ep*ss\sigma + 0.125*Es**2 + 0.25*Es*pp\pi + 0.25*Es*ps\sigma -
0.5*Es*ss\sigma + 0.125*pp\pi**2 + 0.25*pp\pi*ps\sigma - 0.5*pp\pi*ss\sigma +
0.125*ps\sigma**2 - 0.5*ps\sigma*ss\sigma + 1.0*sp\sigma**2 + 0.5*ss\sigma**2):
1,
 0.5*Ep + 0.5*Es - 0.5*pp\pi - 0.5*ps\sigma - 1.0*ss\sigma +
1.4142135623731*sqrt(0.125*Ep**2 - 0.25*Ep*Es - 0.25*Ep*pp\pi - 0.25*Ep*ps\sigma
+ 0.5*Ep*ss\sigma + 0.125*Es**2 + 0.25*Es*pp\pi + 0.25*Es*ps\sigma -
0.5*Es*ss\sigma + 0.125*pp\pi**2 + 0.25*pp\pi*ps\sigma - 0.5*pp\pi*ss\sigma +
0.125*ps\sigma**2 - 0.5*ps\sigma*ss\sigma + 1.0*sp\sigma**2 + 0.5*ss\sigma**2):
1,
 1.0*Ep + 1.0*pp\pi + 1.0*ps\sigma: 1}
```

[ ]:
```
# I disable solving analytical form of eigenvectors as it requires a lot of time
# H.eigenvects()
```

[ ]:
```
r = symbols("r", real=True)
n = 2.00
nc = 6.5
rc = 2.18
ro = 1.536
betar = (ro / r) ** n * exp(n * (-((r / rc) ** nc) + (ro / rc) ** nc))
betar
```

[ ]:

$$2.89726083927909 \left(\frac{1}{r}\right)^{2.0} e^{-0.0126200997390625 r^{6.5}}$$

[ ]:
```
Hr = H
Hr = Hr.subs(
    [
```

```
        (Es, -2.99),
        (Ep, 3.71),
        (sss, -5.00 * betar),
        (sps, 4.70 * betar),
        (pps, 5.50 * betar),
        (ppp, -1.55 * betar),
    ]
)
```

[ ]:

```
Hsub = Hr
Hsub = Hr.subs(r, 1.2)
Hsub
```

[ ]:

$$\left[\begin{matrix} -22.2962256801866 & 9.07392606968772 & 9.07392606968772 & 0 & 0 & 0 \\ 9.07392606968772 & 3.71 & 7.62595914367372 & 0 & 0 & 0 \\ 9.07392606968772 & 7.62595914367372 & 3.71 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3.71 & 0 & 0 \\ 0 & 0 & 0 & 0 & -2.99 & -9.07392606968772i \\ 0 & 0 & 0 & 0 & 9.07392606968772i & 3.71 \\ 0 & 0 & 0 & 0 & -9.07392606968772i & -13.6108891045316i \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}\right.$$

[ ]:

```
eigvals = Hsub.eigenvals()
eigvals
```

[ ]:

```
{-26.6332236486924: 1,
 -3.91595914367372: 1,
 15.6729571121795: 1,
 3.71000000000000: 4,
 -16.9952307469181 - 6.63604414520936e-64*I: 1,
 0.273091502994089 - 1.34014039875119e-64*I: 1,
 21.152139243924 - 2.36068562130255e-63*I: 1,
 -10.1402337509701: 1,
 22.5405002874830: 1,
 11.3359591436737: 1,
 -16.9952307469181 + 6.63604414520936e-64*I: 1,
 0.273091502994089 + 1.34014039875119e-64*I: 1,
 21.152139243924 + 2.36068562130255e-63*I: 1}
```

[ ]:
```
eigs = Hsub.eigenvects()
eigs = sorted(eigs, key=lambda e: re(e[0]))
# energies = [e[0].real if isinstance(e[0], complex) else e[0] for e in eigs]
energies = [re(e[0]) for e in eigs]
energies
```

[ ]:
```
[-26.6332236486924,
 -16.9952307469181,
 -16.9952307469181,
 -10.1402337509701,
 -3.91595914367372,
 0.273091502994089,
 0.273091502994089,
 3.71000000000000,
 3.71000000000000,
 3.71000000000000,
 3.71000000000000,
 11.3359591436737,
 15.6729571121795,
 21.1521392439240,
 21.1521392439240,
 22.5405002874830]
```

[ ]:
```
eigfuncs = [Phi*e[2][0] for e in eigs]
eigfuncs = [e.applyfunc(lambda i: qapply(expand(i)).doit())[0] for e in␣
↪eigfuncs]
eigfuncs
```

[ ]:
```
[0.113200333291455*|PAx> + 0.113200333291455*|PAy> + 0.113200333291455*|PBx> +
0.113200333291455*|PBy> + 0.113200333291455*|PCx> + 0.113200333291455*|PCy> +
0.113200333291455*|PDx> + 0.113200333291455*|PDy> - 0.473678550375048*|SA> -
0.473678550375048*|SB> - 0.473678550375048*|SC> - 0.473678550375048*|SD>,
 -0.0222319506230092*|PAx> - 0.279838553479688*I*|PAx> + 0.265714228591834*|PAy>
+ 0.0905528815993989*I*|PAy> - 0.0905528815993989*|PBx> +
0.265714228591834*I*|PBx> + 0.279838553479688*|PBy> - 0.0222319506230092*I*|PBy>
+ 0.0222319506230092*|PCx> + 0.279838553479688*I*|PCx> - 0.265714228591834*|PCy>
- 0.0905528815993989*I*|PCy> + 0.0905528815993989*|PDx> -
0.265714228591834*I*|PDx> - 0.279838553479688*|PDy> + 0.0222319506230092*I*|PDy>
+ 0.239974946467251*|SA> - 0.186559035653125*I*|SA> + 0.186559035653125*|SB> +
0.239974946467251*I*|SB> - 0.239974946467251*|SC> + 0.186559035653125*I*|SC> -
0.186559035653125*|SD> - 0.239974946467251*I*|SD>,
 -0.0222319506230092*|PAx> + 0.279838553479688*I*|PAx> + 0.265714228591834*|PAy>
- 0.0905528815993989*I*|PAy> - 0.0905528815993989*|PBx> -
0.265714228591834*I*|PBx> + 0.279838553479688*|PBy> + 0.0222319506230092*I*|PBy>
+ 0.0222319506230092*|PCx> - 0.279838553479688*I*|PCx> - 0.265714228591834*|PCy>
+ 0.0905528815993989*I*|PCy> + 0.0905528815993989*|PDx> +
0.265714228591834*I*|PDx> - 0.279838553479688*|PDy> - 0.0222319506230092*I*|PDy>
```

+ 0.239974946467251*|SA> + 0.186559035653125*I*|SA> + 0.186559035653125*|SB> -
0.239974946467251*I*|SB> - 0.239974946467251*|SC> - 0.186559035653125*I*|SC> -
0.186559035653125*|SD> + 0.239974946467251*I*|SD>,
0.318108275324079*|PAx> + 0.318108275324079*|PAy> - 0.318108275324079*|PBx> -
0.318108275324079*|PBy> + 0.318108275324079*|PCx> + 0.318108275324079*|PCy> -
0.318108275324079*|PDx> - 0.318108275324079*|PDy> + 0.218206898013515*|SA> -
0.218206898013515*|SB> + 0.218206898013515*|SC> - 0.218206898013515*|SD>,
0.353553390593274*|PAx> - 0.353553390593274*|PAy> - 0.353553390593274*|PBx> +
0.353553390593274*|PBy> + 0.353553390593274*|PCx> - 0.353553390593274*|PCy> -
0.353553390593274*|PDx> + 0.353553390593274*|PDy> - 2.01094893482639e-65*|SA> -
2.01094893482639e-65*|SB> - 2.01094893482639e-65*|SC> -
2.01094893482639e-65*|SD>,
-0.251180229586438*|PAx> + 0.042143989758508*I*|PAx> - 0.241076577001577*|PAy>
- 0.0821566042936517*I*|PAy> + 0.0821566042936517*|PBx> -
0.241076577001577*I*|PBx> - 0.042143989758508*|PBy> - 0.251180229586438*I*|PBy>
+ 0.251180229586438*|PCx> - 0.042143989758508*I*|PCx> + 0.241076577001577*|PCy>
+ 0.0821566042936517*I*|PCy> - 0.0821566042936517*|PDx> +
0.241076577001577*I*|PDx> + 0.042143989758508*|PDy> + 0.251180229586438*I*|PDy>
+ 0.345652090912145*|SA> + 0.0280959931723387*I*|SA> - 0.0280959931723387*|SB> +
0.345652090912145*I*|SB> - 0.345652090912145*|SC> - 0.0280959931723387*I*|SC> +
0.0280959931723387*|SD> - 0.345652090912145*I*|SD>,
-0.251180229586438*|PAx> - 0.042143989758508*I*|PAx> - 0.241076577001577*|PAy>
+ 0.0821566042936517*I*|PAy> + 0.0821566042936517*|PBx> +
0.241076577001577*I*|PBx> - 0.042143989758508*|PBy> + 0.251180229586438*I*|PBy>
+ 0.251180229586438*|PCx> + 0.042143989758508*I*|PCx> + 0.241076577001577*|PCy>
- 0.0821566042936517*I*|PCy> - 0.0821566042936517*|PDx> -
0.241076577001577*I*|PDx> + 0.042143989758508*|PDy> - 0.251180229586438*I*|PDy>
+ 0.345652090912145*|SA> - 0.0280959931723387*I*|SA> - 0.0280959931723387*|SB> -
0.345652090912145*I*|SB> - 0.345652090912145*|SC> + 0.0280959931723387*I*|SC> +
0.0280959931723387*|SD> + 0.345652090912145*I*|SD>,
0.5*|PAz> + 0.5*|PBz> + 0.5*|PCz> + 0.5*|PDz>,
0.5*|PAz> + 0.5*I*|PBz> - 0.5*|PCz> - 0.5*I*|PDz>,
0.5*|PAz> - 0.5*|PBz> + 0.5*|PCz> - 0.5*|PDz>,
0.5*|PAz> - 0.5*I*|PBz> - 0.5*|PCz> + 0.5*I*|PDz>,
-0.353553390593274*|PAx> + 0.353553390593274*|PAy> - 0.353553390593274*|PBx> +
0.353553390593274*|PBy> - 0.353553390593274*|PCx> + 0.353553390593274*|PCy> -
0.353553390593274*|PDx> + 0.353553390593274*|PDy> - 3.62672388511659e-64*|SA> +
3.62672388511659e-64*|SB> - 3.62672388511659e-64*|SC> +
3.62672388511659e-64*|SD>,
0.33494131507281*|PAx> + 0.33494131507281*|PAy> + 0.33494131507281*|PBx> +
0.33494131507281*|PBy> + 0.33494131507281*|PCx> + 0.33494131507281*|PCy> +
0.33494131507281*|PDx> + 0.33494131507281*|PDy> + 0.160089446605931*|SA> +
0.160089446605931*|SB> + 0.160089446605931*|SC> + 0.160089446605931*|SD>,
0.324959802169487*|PAx> - 0.0270116220171825*I*|PAx> - 0.105184883861477*|PAy>
- 0.308649705907974*I*|PAy> + 0.308649705907974*|PBx> -
0.105184883861477*I*|PBx> + 0.0270116220171825*|PBy> + 0.324959802169487*I*|PBy>
- 0.324959802169487*|PCx> + 0.0270116220171825*I*|PCx> + 0.105184883861477*|PCy>

```
+ 0.308649705907974*I*|PCy> - 0.308649705907974*|PDx> +
0.105184883861477*I*|PDx> - 0.0270116220171825*|PDy> - 0.324959802169487*I*|PDy>
+ 0.105854875817466*|SA> - 0.161671716034706*I*|SA> + 0.161671716034706*|SB> +
0.105854875817466*I*|SB> - 0.105854875817466*|SC> + 0.161671716034706*I*|SC> -
0.161671716034706*|SD> - 0.105854875817466*I*|SD>,
 0.324959802169487*|PAx> + 0.0270116220171825*I*|PAx> - 0.105184883861477*|PAy>
+ 0.308649705907974*I*|PAy> + 0.308649705907974*|PBx> +
0.105184883861477*I*|PBx> + 0.0270116220171825*|PBy> - 0.324959802169487*I*|PBy>
- 0.324959802169487*|PCx> - 0.0270116220171825*I*|PCx> + 0.105184883861477*|PCy>
- 0.308649705907974*I*|PCy> - 0.308649705907974*|PDx> -
0.105184883861477*I*|PDx> - 0.0270116220171825*|PDy> + 0.324959802169487*I*|PDy>
+ 0.105854875817466*|SA> + 0.161671716034706*I*|SA> + 0.161671716034706*|SB> -
0.105854875817466*I*|SB> - 0.105854875817466*|SC> - 0.161671716034706*I*|SC> -
0.161671716034706*|SD> + 0.105854875817466*I*|SD>,
 0.154295577287038*|PAx> + 0.154295577287038*|PAy> - 0.154295577287038*|PBx> -
0.154295577287038*|PBy> + 0.154295577287038*|PCx> + 0.154295577287038*|PCy> -
0.154295577287038*|PDx> - 0.154295577287038*|PDy> - 0.449873037266427*|SA> +
0.449873037266427*|SB> - 0.449873037266427*|SC> + 0.449873037266427*|SD>]
```

0.4 (4) Electronic energy

As there are total $4 \times 4 = 16$ electrons involving the bond formation, the lowest $16/2 = 8$ orbitals will be occupied to lower the total energy. Therefore, the change in electronic energy upon forming the molecule of bond length 1.2 will be

[ ]:
```python
sum(energies[: int(4 * 4 / 2)])*2 # eV
```

[ ]:

-140.847390062368

[ ]:
```python

```