Chapter 1

数学の関係式

ここでは，ベクトル空間と内積空間についての定義を与えたあと，Hermite 行列が Herm(N) が内積空間となることを示す．ただし，Hermite 行列は，

$$\mathrm{Herm}(N) \in \left\{ \hat{H} \in \mathbb{C}^{N\times N} \mid \hat{H} = \hat{H}^\dagger \right\} \tag{0.1}$$

である．

V が K 上のベクトル空間であることは，以下の条件を全て満たすことである．ただし，$\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w} \in V$，$a, b \in K$ とする．

ベクトル空間の定義

1. $\forall \boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w} \in V\ \boldsymbol{u} + (\boldsymbol{v} + \boldsymbol{v}) = (\boldsymbol{u} + \boldsymbol{v}) + \boldsymbol{v}$
2. $\exists \mathbf{0} \in V\ \forall \boldsymbol{v}\ \boldsymbol{u} + \mathbf{0} = \mathbf{0} + \boldsymbol{u} = \boldsymbol{u}$
3. $\forall \boldsymbol{v} \in V\ \exists - \boldsymbol{u}\ \boldsymbol{u} + (-\boldsymbol{u}) = \mathbf{0}$
4. $\forall \boldsymbol{u}, \boldsymbol{v} \in V\ \boldsymbol{u} + \boldsymbol{v} = \boldsymbol{v} + \boldsymbol{u}$
5. $\forall a \in K\ \forall \boldsymbol{u}, \boldsymbol{v} \in V\ a(\boldsymbol{u} + \boldsymbol{v}) = a\boldsymbol{u} + a\boldsymbol{v}$
6. $\forall a, b \in K\ \forall \boldsymbol{v} \in V\ (a + b)\boldsymbol{v} = a\boldsymbol{v} + b\boldsymbol{v}$
7. $\forall a, b \in K\ \forall \boldsymbol{v} \in V\ a(b\boldsymbol{v}) = (ab)\boldsymbol{v}$
8. $\exists 1 \in K\ \forall \boldsymbol{v} \in V\ 1\boldsymbol{v} = \boldsymbol{v}$

Herm(N) は通常の行列の演算規則に従えば，明らかに $\mathbb{C}$ 上のベクトル空間である．

内積は，ベクトル空間 V に対して定義された演算 $(\cdot, \cdot) : V \times V \to \mathbb{C}$ が以下の性質を満たすものである．ただし，$\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w} \in V$，$a, b \in \mathbb{C}$ とする．

内積の定義

1. $(\boldsymbol{u}, \lambda\boldsymbol{v}) = \lambda(\boldsymbol{u}, \boldsymbol{v})$
2. $(\boldsymbol{u}, \boldsymbol{v}) = (\boldsymbol{v}, \boldsymbol{u})^*$
3. $\forall \boldsymbol{u}\ (\boldsymbol{u}, \boldsymbol{u}) \geq 0$
4. $(\boldsymbol{u}, \boldsymbol{u}) = 0 \implies \boldsymbol{u} = \mathbf{0}$

$A, B \in \mathrm{Herm}(N)$ に対して内積を定義するには，対角和を用いて，

$$(A, B) = \mathrm{tr}\big(A^\dagger B\big) \tag{0.2}$$

と定義すればよい．行列の対角和が Herm(N) の内積になることは非自明なので示す．ただし，$A, B \in \mathrm{Herm}(N) A$ の固有値を $\lambda_1, \ldots \lambda_N$ とする．

Proof. 1. $(A, \lambda B) = \mathrm{tr}\left(A^\dagger \lambda B\right) = \lambda\, \mathrm{tr}\left(A^\dagger B\right) = \lambda(A, B)$

2. $(B, A)^* = \mathrm{tr}\left(B^\dagger A\right)^* = \mathrm{tr}\left(\left(B^\dagger A\right)^\dagger\right) = \mathrm{tr}\left\{A^\dagger B\right\} = (A, B)$

3. A は Hermite 行列であるから固有値は全て実数で，$(A, A) = \mathrm{tr}\left\{A^\dagger A\right\} = \mathrm{tr}\left\{A^2\right\} = \sum_{i=1}^{N} \lambda_i^2 \geq 0$

4. $(A, A) = \sum_{i=1}^{N} \lambda_i^2 = 0$ となるのは $\lambda_1 = \cdots = \lambda_N = 0$ となるときのみで，そのときは A は 0 行列である．

よって，対角和を用いて内積を定義すると，Herm(N) は内積空間になる． □

Pauli 行列を $\sqrt{2}$ で割ったものは Herm(2) の正規直交基底となる．ただし，Pauli 行列は,

$$\hat{\sigma}^0 \coloneqq \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{0.3}$$

$$\hat{\sigma}^1 \coloneqq \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{0.4}$$

$$\hat{\sigma}^2 \coloneqq \begin{pmatrix} 0 & -\mathrm{i} \\ \mathrm{i} & 0 \end{pmatrix} \tag{0.5}$$

$$\hat{\sigma}^3 \coloneqq \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{0.6}$$

と定義される．まず，Pauli 行列を $\sqrt{2}$ で割ったものが Herm(2) の基底であることを示す．

Proof. $a_0, a_1, a_2, a_3 \in \mathbb{C}$ を用いると，

$$\frac{1}{\sqrt{2}} a_0 \hat{\sigma}^0 + \frac{1}{\sqrt{2}} a_1 \hat{\sigma}^1 + \frac{1}{\sqrt{2}} a_2 \hat{\sigma}^2 + \frac{1}{\sqrt{2}} a_3 \hat{\sigma}^3 = \frac{1}{\sqrt{2}} \begin{pmatrix} a_0 + a_3 & a_1 - \mathrm{i}a_2 \\ a_1 + \mathrm{i}a_2 & a_0 - a_3 \end{pmatrix} \tag{0.7}$$

となり，Pauli 行列を $\sqrt{2}$ で割ったものの線形結合で任意の Hermite 行列が書けることが分かる． □

また，Pauli 行列を $\sqrt{2}$ で割ったものは正規直交基底を成すことが分かる．

Proof. Pauli 行列同士の内積について，

$$\left(\frac{1}{\sqrt{2}}\hat{\sigma}^0, \frac{1}{\sqrt{2}}\hat{\sigma}^0\right) = \frac{1}{2}\mathrm{tr}\left\{\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\right\} = \frac{1}{2}\mathrm{tr}\left\{\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\right\} = 1 \tag{0.8}$$

$$\left(\frac{1}{\sqrt{2}}\hat{\sigma}^1, \frac{1}{\sqrt{2}}\hat{\sigma}^1\right) = \frac{1}{2}\mathrm{tr}\left\{\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}\right\} = \frac{1}{2}\mathrm{tr}\left\{\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\right\} = 1 \tag{0.9}$$

$$\left(\frac{1}{\sqrt{2}}\hat{\sigma}^2, \frac{1}{\sqrt{2}}\hat{\sigma}^2\right) = \frac{1}{2}\mathrm{tr}\left\{\begin{pmatrix} 0 & -\mathrm{i} \\ \mathrm{i} & 0 \end{pmatrix}\begin{pmatrix} 0 & -\mathrm{i} \\ \mathrm{i} & 0 \end{pmatrix}\right\} = \frac{1}{2}\mathrm{tr}\left\{\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\right\} = 1 \tag{0.10}$$

$$\left(\frac{1}{\sqrt{2}}\hat{\sigma}^3, \frac{1}{\sqrt{2}}\hat{\sigma}^3\right) = \frac{1}{2}\mathrm{tr}\left\{\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}\right\} = \frac{1}{2}\mathrm{tr}\left\{\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\right\} = 1 \tag{0.11}$$

$$\left(\frac{1}{\sqrt{2}}\hat{\sigma}^0, \frac{1}{\sqrt{2}}\hat{\sigma}^1\right) = \frac{1}{2}\mathrm{tr}\left\{\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}\right\} = \frac{1}{2}\mathrm{tr}\left\{\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}\right\} = 0 \tag{0.12}$$

$$\left(\frac{1}{\sqrt{2}}\hat{\sigma}^0, \frac{1}{\sqrt{2}}\hat{\sigma}^2\right) = \frac{1}{2}\mathrm{tr}\left\{\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} 0 & -\mathrm{i} \\ \mathrm{i} & 0 \end{pmatrix}\right\} = \frac{1}{2}\mathrm{tr}\left\{\begin{pmatrix} 0 & -\mathrm{i} \\ \mathrm{i} & 0 \end{pmatrix}\right\} = 0 \tag{0.13}$$

$$\left(\frac{1}{\sqrt{2}}\hat{\sigma}^0, \frac{1}{\sqrt{2}}\hat{\sigma}^3\right) = \frac{1}{2}\mathrm{tr}\left\{\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}\right\} = \frac{1}{2}\mathrm{tr}\left\{\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}\right\} = 0 \tag{0.14}$$

$$\left(\frac{1}{\sqrt{2}}\hat{\sigma}^1, \frac{1}{\sqrt{2}}\hat{\sigma}^2\right) = \frac{1}{2}\mathrm{tr}\left\{\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} 0 & -\mathrm{i} \\ \mathrm{i} & 0 \end{pmatrix}\right\} = \frac{1}{2}\mathrm{tr}\left\{\begin{pmatrix} \mathrm{i} & 0 \\ 0 & -\mathrm{i} \end{pmatrix}\right\} = 0 \tag{0.15}$$

$$\left(\frac{1}{\sqrt{2}}\hat{\sigma}^1, \frac{1}{\sqrt{2}}\hat{\sigma}^3\right) = \frac{1}{2}\mathrm{tr}\left\{\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}\right\} = \frac{1}{2}\mathrm{tr}\left\{\begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}\right\} = 0 \tag{0.16}$$

$$\left(\frac{1}{\sqrt{2}}\hat{\sigma}^2, \frac{1}{\sqrt{2}}\hat{\sigma}^3\right) = \frac{1}{2}\mathrm{tr}\left\{\begin{pmatrix} 0 & -\mathrm{i} \\ \mathrm{i} & 0 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}\right\} = \frac{1}{2}\mathrm{tr}\left\{\begin{pmatrix} 0 & \mathrm{i} \\ \mathrm{i} & 0 \end{pmatrix}\right\} = 0 \tag{0.17}$$

となり，Pauli 行列は正規直交基底であると分かる．内積の順序入れ替えについては，以上の計算結果が全て実数であることから省略する． □

Chapter 2

ソースコード

qst-code

```
import matplotlib.pyplot as plt
import numpy as np
import mpl_toolkits
from matplotlib import cm
from matplotlib.colors import Normalize
import pandas as pd

class HVDR:
    def __init__(self, l):
        self.l = l
    def __mul__(self, other):
        ret = list()
        for x in self.l:
            tmp = list()
            for y in other.l:
                tmp.append(x * y)
            ret.append(tmp)
        return ret

def calc_sigma_pow():
    sigma_0 = HVDR([1, 1, 0, 0])
    sigma_1 = HVDR([-1, -1, 2, 0])
    sigma_2 = HVDR([-1, -1, 0, 2])
    sigma_3 = HVDR([1, -1, 0, 0])
    sigma = [sigma_0, sigma_1, sigma_2, sigma_3]
    sigma_pow = [[None for _ in range(4)] for _ in range(4)]
    for i in range(4):
        for j in range(4):
            sigma_pow[i][j] = sigma[i] * sigma[j]
    return sigma_pow
def load_data(path):
    polar_dict = {'H': 0, 'V': 1, 'D': 2, 'R': 3}
    raw_data = np.loadtxt(path, dtype = "str", delimiter = ',')
    data = [[None for _ in range(4)] for _ in range(4)]
    for i in range(1, len(raw_data)):
        data[polar_dict[raw_data[i][0]]][polar_dict[raw_data[i][1]]] = int(raw_data[i][2])
    return data
def calc_uij(sigma_pow, data):
    u = [[0 for _ in range(4)] for _ in range(4)]
    for i1 in range(4):
        for j1 in range(4):
            for i2 in range(4):
```

```python
                for j2 in range(4):
                    u[i1][j1] += sigma_pow[i1][j1][i2][j2] * data[i2][j2]
    return u
def calc_sigma_matrix_tensor():
    sigma_num_0 = np.array([[1, 0], [0, 1]])
    sigma_num_1 = np.array([[0, 1], [1, 0]])
    sigma_num_2 = np.array([[0, -1j], [1j, 0]])
    sigma_num_3 = np.array([[1, 0], [0, -1]])
    sigma_num = [sigma_num_0, sigma_num_1, sigma_num_2, sigma_num_3]
    sigma_num_pow = [[None for _ in range(4)] for _ in range(4)]
    for i in range(4):
        for j in range(4):
            sigma_num_pow[i][j] = np.kron(sigma_num[i], sigma_num[j])
    return sigma_num_pow
def estimate_rho(u, sigma_num_pow):
    rho = [[0 for _ in range(4)] for _ in range(4)]
    rho_trace = 0
    for i1 in range(4):
        for j1 in range(4):
            for i2 in range(4):
                for j2 in range(4):
                    rho[i2][j2] += u[i1][j1] * sigma_num_pow[i1][j1][i2][j2]
    for i in range(4): rho_trace += rho[i][i]
    rho_norm = [[rho[i][j] / rho_trace for j in range(4)] for i in range(4)]
    rho_norm = np.array(rho_norm)
    return rho_norm
def make_graph(rho_norm, data_name):
    rho_real_imag = [rho_norm.real, rho_norm.imag]
    tmp_x = np.arange(4)
    tmp_y = np.arange(4)
    tmp_X, tmp_Y = np.meshgrid(tmp_x, tmp_y)
    label_bra = [
        r"$\left| \rm{HH} \right\rangle$",
        r"$\left| \rm{VH} \right\rangle$",
        r"$\left| \rm{HV} \right\rangle$",
        r"$\left| \rm{VV} \right\rangle$"
    ]
    label_ket = [
        r"$\left\langle \rm{HH} \right|$",
        r"$\left\langle \rm{VH} \right|$",
        r"$\left\langle \rm{HV} \right|$",
        r"$\left\langle \rm{VV} \right|$"
    ]
    x = tmp_X.ravel()
    y = tmp_Y.ravel()
    z = np.zeros_like(x)
    dx = dy = 0.5
    for i in range(2):
        fig = plt.figure()
        ax = fig.add_subplot(111, projection="3d")
        dz = rho_real_imag[i].ravel()
        norm = Normalize(vmin=-1, vmax=1)
        colors = cm.coolwarm_r(norm(dz))
        alpha = 0.7
        colors[:, 3] = alpha
        ax.bar3d(x, y, z, dx, dy, dz, color=colors, shade=True)
        ax.set_xticks(tmp_x + dx / 2)
        ax.set_xticklabels(label_bra, ha="center")
```

```
        ax.set_yticks(tmp_y + dy / 2)
        ax.set_yticklabels(label_ket, ha="center")
        ax.set_zlim(-0.75, 0.75)
        ax.zaxis.set_tick_params(labelleft=False, labelright=False, labeltop=False, labelbottom=False)
        mappable = cm.ScalarMappable(norm=norm, cmap="coolwarm_r")
        mappable.set_array(dz)
        fig.colorbar(mappable, ax=ax, shrink=0.6, aspect=10, label=r"$\hat{\rho}$")
        title = data_name + "_" + "riemaalg"[i:: 2]
        ax.set_title(title)
        title += ".pdf"
        plt.savefig(title, bbox_inches = "tight")
def main():
    sigma_pow = calc_sigma_pow()
    sigma_num_pow = calc_sigma_matrix_tensor()
    datas = ["data1", "data2"]
    for i in range(2):
        data = load_data(datas[i] + ".csv")
        u = calc_uij(sigma_pow, data)
        rho_norm = estimate_rho(u, sigma_num_pow)
        make_graph(rho_norm, datas[i])
if (__name__ == "__main__"):
    main()
```