Matrix problems

行列問題の解法

Fundamental matrix operations

C = A+B:

```
for ix in range(nx):
    for iy in range(ny):
        c[ix][iy] = a[ix][iy] + b[ix][iy];
```

C = AB:

```
for ix in range(nx):
    for iy in range(ny):
        c[ix][iy] = 0.0;
        for k in range(nk):
            c[ix][iy] = c[ix][iy] + a[ix][k]*b[k][iy];
```

To solve BC = A

(i) Directly solve BC = A

(ii) B^{-1} can be obtained to calculate $B^{-1}A$, but this should be avoided if possible.

=> Better to use algebra libraries

Gauss elimination method (Gaussの消去法)

Upon a square matrix (正方行列) A and a vector B are given,
solution of $\boldsymbol{AX = B}$ is obtained by $X = A^{-1}B$.

・Efficient for case more than one solutions for the same A and different B.

・Can produce roundoff errors and not efficient

=> Solve the linear simultaneous equations directly.

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & & a_{2n} \\ a_{31} & a_{32} & a_{33} & & a_{3n} \\ \vdots & & & \ddots & \\ a_{n1} & a_{n2} & a_{n3} & & a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$$

Multiply a_{i1}/a_{11} ($i = 2, 3, \ldots, n$) to the first line and subtract it from i-th line

=> make all a_{i1} ($i \geq 2$) zero.

Repeat this procedure for all the lines, A will be converted to upper-right triangle matrix (右上三角行列)

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ 0 & a_{22}' & a_{23}' & \cdots & a_{2n}' \\ 0 & 0 & a_{33}' & & a_{3n}' \\ \vdots & 0 & 0 & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & a_{nn}' \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1' \\ b_2' \\ \vdots \\ b_n' \end{pmatrix}$$

Solve from the last line to upper lines, giving all x_i

Note: Converting A to a band or triangle matrix enables solve the equation very easy

Row reduction method (掃き出し法)

Similar to the Gauss elimination method, but eliminates all non-diagonal terms

$$\begin{pmatrix} a_{11} & 0 & 0 & \cdots & 0 \\ 0 & a_{22}' & 0 & \cdots & 0 \\ 0 & 0 & a_{33}' & & 0 \\ \vdots & 0 & 0 & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & a_{nn}' \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1' \\ b_2' \\ \vdots \\ b_n' \end{pmatrix}$$

Obtain the solution by $x_i = b_i' / a_{ii}'$

Important: Regular matrix can be converted to triangle / band matrixes
(正則行列は、適当な行列による変換で三角行列や帯行列に分解できる)

=> *ex.* *LU* decomposition (LU分解)**: $A = LU$**

***L*: Left-lower triangle matrix, *U*: Right-upper triangle matrix**

Solution of linear simul. eqs. : LU decomposition

1. Convert $AX = B$ to $LUX = B$ by $A = LU$

2. Solve $LY = B$ to obtain Y

3. Solve $UX = Y$ to obtain X

Diagonalization of real symmetric matrix: Jacobi method (ヤコビ法)

Diagonalization of $A = \begin{pmatrix} a_{11} & a_{12} \\ a_{12} & a_{22} \end{pmatrix}$

=> can be done by conversion $U^{\mathrm{T}}AU$ with an orthogonal matrix (直交行列) U

$$U = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$$

$$U^T AU = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} a_{11} & a_{12} \\ a_{12} & a_{22} \end{pmatrix} \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$$

$$= \begin{pmatrix} a_{11}\cos^2\theta + 2a_{12}\cos\theta\sin\theta + a_{22}\sin^2\theta & (-a_{11}+a_{22})\cos\theta\sin\theta + a_{12}(\cos^2\theta - \sin^2\theta) \\ (-a_{11}+a_{22})\cos\theta\sin\theta + a_{12}(\cos^2\theta - \sin^2\theta) & a_{11}\sin^2\theta - 2a_{12}\cos\theta\sin\theta + a_{22}\cos^2\theta \end{pmatrix}$$

$$(-a_{11}+a_{22})\cos\theta\sin\theta + a_{12}(\cos^2\theta - \sin^2\theta) = 1/2\left[(-a_{11}+a_{22})\sin 2\theta + a_{12}\cos 2\theta\right] = 0$$

$$\Rightarrow \quad \theta = \pi/4 \qquad a_{11} = a_{22}$$

$$\theta = (1/2)\tan^{-1}(2a_{12}/(a_{11}-a_{22})) \qquad a_{11} \neq a_{22}$$

Jacobi method

1. Choose the largest absolute value non-diagonal element a_{ij} in

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{12} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{13} & a_{23} & a_{33} & & a_{3n} \\ \vdots & & & \ddots & \vdots \\ a_{1n} & a_{2n} & & \cdots & a_{nn} \end{pmatrix}$$

2. Converting by $A' = U^{T}AU$ with

$$U = \begin{pmatrix} 1 & 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ 0 & 1 & & & & & \vdots \\ \vdots & & \cos\theta & & -\sin\theta & & \vdots \\ \vdots & & & 1 & & & \vdots \\ \vdots & & \sin\theta & & \cos\theta & & \vdots \\ \vdots & & & & & \ddots & 0 \\ 0 & \cdots & \cdots & \cdots & \cdots & 0 & 1 \end{pmatrix}$$

will give $a_{ij}' = 0$

3. Choose the largest absolute value element a_{ij}' and repeat 2

=> The square sum of non-diagonal elements is reduce by a factor of $2a_{ij}^2$

=> finite iterations will complete the diagonalization

But it is hard to estimate the number of iterations required, and Jacobi method is not efficient for a large-size materix

Diagonalization of large-size matrix

Householder method

1. **Convert a symmetric matrix A to a tridiagonal matrix** (三重対角行列) D **using an orthogonal matrix** (直交行列) U

   Note: eigen values of U^TAU are equal to those of A

2. **Solve eigen values of D by bisection method**

QR method

1. **Regular $n \times n$ matrix A is decomposed to $A = QR$** (QR分解) **using a regular orthogonal matrix Q and a right-upper matrix with positive diagonal elements R.**
2. **QR-decompose A_k: $A_k = Q_k R_k$**
3. **Convert A_k to $A_{k+1} = Q_k^T A_k Q_k = R_k Q_k$ (similar transformation,** 相似変換**)**
4. **Repeating 2 and 3 will converge A_k to a right-upper triangle matrix A_R => Solve eigen values of A_R**

   If A is a symmetric matrix, A_R will be a diagonal matrix.

Linear algebra libraries

(線形代数・行列計算ライブラリ)

Fortran, C, C++, etc

LAPACK (Linear Algebra PACKage)

ScaLAPACK (Scalable LAPACK)

Intel Math Kernel Library (MKL)

One API: https://www.intel.com/content/www/us/en/developer/tools/oneapi/overview.html

Python: numpy.linalg, scipy.linalg

matrix.py

Product of matrixes	**AB**	: C	= A @ B
Inner product	**V1·V2**	: inner	= numpy.dot(V1, V2)
		inner	= numpy.inner(V1, V2)
Cross product	**V1 × V2**	: V3	= numpy.cross(V1, V2)
Outer product		: V3	= numpy.outer(V1, V2)
Inverse matrix		: Ai	= numpy.linalg.inv(A)
Determinant		: det	= numpy.linalg.det(A)
Eigen values/vectors		: lA, vA	= numpy.linalg.eig(A)
Solve simul. linear eqs.	**AX = B**	: X	= numpy.linalg.solve(A, B)
LU decomposition		: P, L, U	= scpy.linalg.lu(A)
Cholesky decomposition	**$A=LL^T$**	: L	= numpy.linalg.cholesky(A)
QR decomposition	**A=QR**	: Q, R	= scipy.linalg.qr(A)

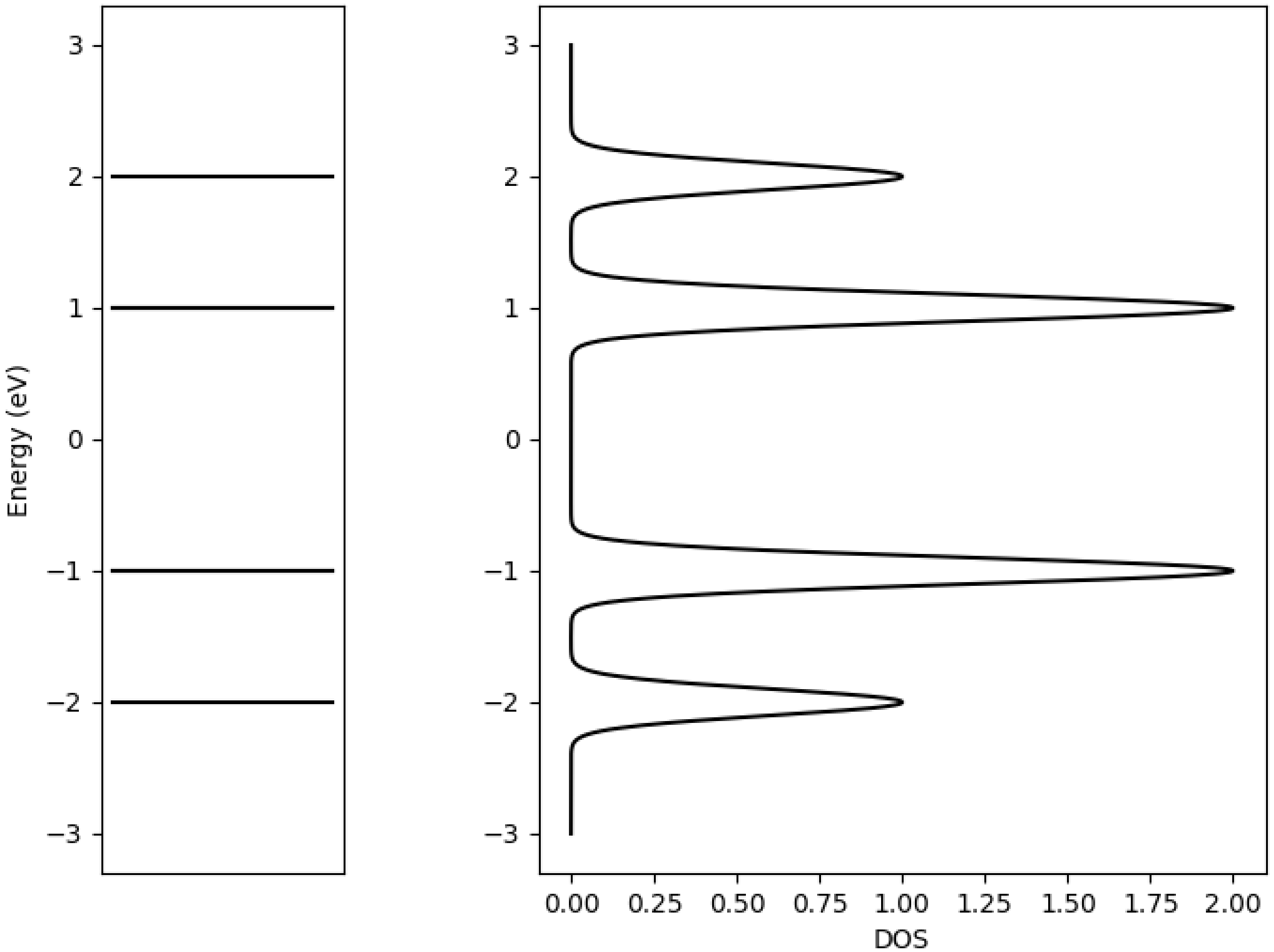
An example of Matrix operation: Electronic structure of benzene

Hückel approximation: benzene.py

Energy level of C 2p: $\varepsilon_{2p} = 0$
Resonance integral: $\beta_{2p\pi\text{-}\pi} = 1$
Diagonalize the following Hamiltonian

```
H = np.array([[0, 1, 0, 0, 0, 1],
              [1, 0, 1, 0, 0, 0],
              [0, 1, 0, 1, 0, 0],
              [0, 0, 1, 0, 1, 0],
              [0, 0, 0, 1, 0, 1],
              [1, 0, 0, 0, 1, 0]])
```

Energy level diagram Density of states



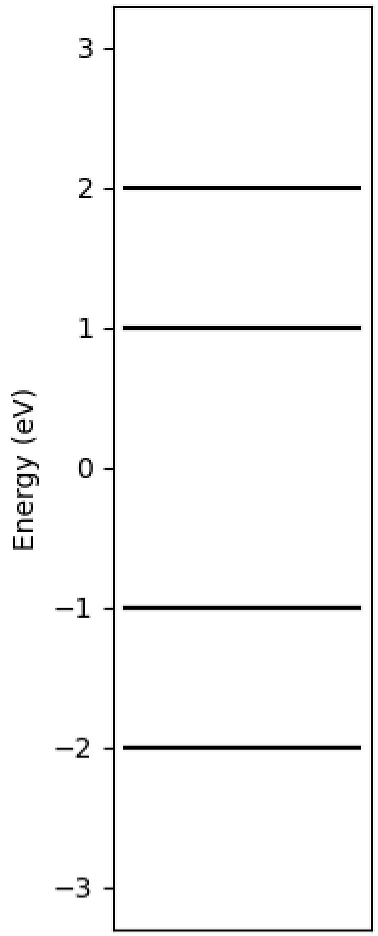
行列計算の応用: ベンゼンの電子準位

Hückel近似を使う: benzene.py

C 2pのエネルギー準位を $\varepsilon_{2p} = 0$
共鳴積分を $\beta_{2p\pi\text{-}\pi} = 1$
としたときの
ハミルトニアンを対角化して
エネルギー準位を計算できる

```
H = np.array([[0, 1, 0, 0, 0, 1],
              [1, 0, 1, 0, 0, 0],
              [0, 1, 0, 1, 0, 0],
              [0, 0, 1, 0, 1, 0],
              [0, 0, 0, 1, 0, 1],
              [1, 0, 0, 0, 1, 0]])
```

エネルギー準位図



状態密度 (Density of states)