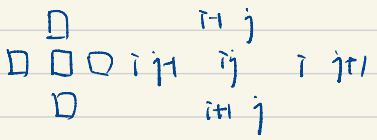
Attached written notes are my pseudo code for the problems.

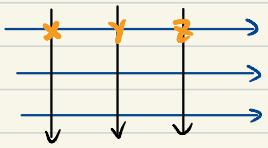
10.9 a>



(j-1 j) (ij j+1)

(i-1 i) (i i+1).

$\begin{pmatrix} a_1 & a_2 \\ b_1 & b_2 \end{pmatrix} \begin{pmatrix} a_1 & a_2 \\ b_1 & b_2 \end{pmatrix}$



(i, j)

j j+1 j+2 ··· N

j j+1 j+2 ··· N

(x+y)+(y+z)
= x+2y+z

```
for i in len(N):
    for j in len(N-1):
        S[i,j] * S[i,j+1]
```

b>

old

-1·1·(-1) = +1

-1	(-1)	1
-1	1	-1
1	-1	1

→

new

-1·1 = -1

-1	(1)	1
-1	1	-1
1	-1	1

-1 - (+1) = -2

2 new.

After calculating the ΔE

```
if ΔE < 0
    config[i][j]  x = -1

if ΔE >=0:
    if random() < e^(-ΔE/kT)
        config[i][j]  x = -1

    else
        config[i][j] x = 1

return (config).
```

c>

```
M = []
def magnetization():
```

```
...   (same as part b)

        m = np.sum(config)
        M.append(m)
        return(M)
```

d) `def animation( ):`

```
...
        img.set_data(config)
```

XY model

$$E = -J \sum \cos(\theta_i - \theta_j)$$

For all rows: `for j in range(N):`

```
        for i in range(N-1):
            energy += cos(config[j,i] - config[j,i+1]
```

same thing for all columns.

MCMC:

```
    for h in (steps):
        i = random
        j = random

        θ = config[i][j]
        θ' = (0, 2π)
        copy = np.copy(config)
        config[i][j] = θ'

        ΔE = energy(config) - energy(copy)

        ...

    same in 10.9
```

```
T = 0.2
print(energy(runmcmc( )))

Energy = [ ]
T_ = np.arange(0.2, 1.6, 0.1)
for T in T_:
        energy(runmcmc(config)))
        Energy.append
```