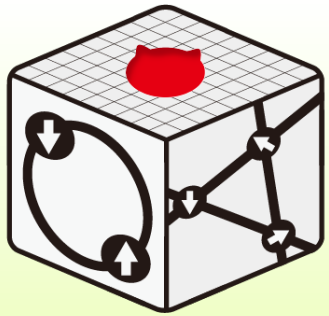
Exercises of mVMC

Takahiro Misawa
BAQIS



mVMC

Basic exercises [demonstration]

1. Heisenberg & Hubbard chain
2. Heisenberg & Hubbard (square lattice)
3. Using Hartree-Fock solutions as initial states

git clone https://github.com/issp-center-dev/mVMC-tutorial.git

Sample scripts: mVMC-tutorial/HandsOn/2022_1128/Samples

Advanced exercise
(Let's generate quantum phases by mVMC)

1. Hubbard + V → Charge order
2. Heisenberg+J2 → stripe magnetic order
3. Attractive Hubbard → superconductivity
4. 1D Kondo lattice → Kondo insulator
5. Data used in researches
[Data repository in ISSP]

1. Calculation flow

Step 1: Optimization [stan_opt.in]

Step 2: Calc. of 1- and 2-body Green functions
for optimized wave functions [stan_aft.in]

Step 3: Calc. of correlations functions such as
spin/charge structure factors
from 1- and 2-body Green functions
[VMCcor.py]

1. Heisenberg chain

$$H = \sum_{\langle i,j \rangle} \vec{S}_i \cdot \vec{S}_j$$

mVMC-tutorial/HandsOn/2022_1128/Samples/1D_Heisenberg

Step 1: Optimization [stan_opt.in]

stan_opt.in

```
L              = 4
Lsub           = 2
lattice        = "chain"
model          = "Spin"
J              = 1.0
2Sz            = 0
NVMCSample     = 200
NSROptItrStep  = 600
NMPTrans       = 1
NSPStot        = 0
```

Exercise: Yellow shaded boxes

```
cp -r 1D_Heisenberg L4_1D_Heisenberg
cd  ./L4_1D_Heisenberg

 vmcdry  stan_opt.in

 vmc ./namelist.def

 gnuplot
   plot ./output/zvo_out_001.dat u 1
```

1. Heisenberg chain

$$H = \sum_{\langle i,j \rangle} \vec{S}_i \cdot \vec{S}_j$$

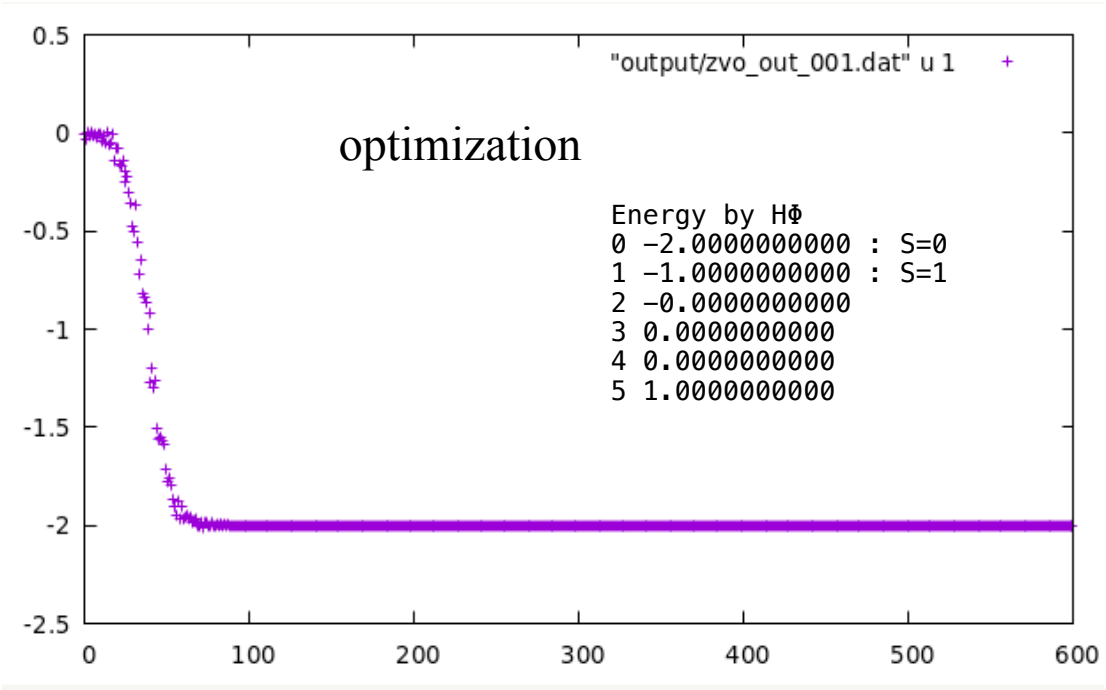
`mVMC-tutorial/HandsOn/2022_1128/Samples/1D_Heisenberg`

Step 1: Optimization [stan_opt.in]

stan_opt.in

```
L              = 4
Lsub           = 2
lattice        = "chain"
model          = "Spin"
J              = 1.0
2Sz            = 0
NVMCSample     = 200
NSROptItrStep  = 600
NMPTrans       = 1
NSPStot        = 0
```

plot "./output/zvo_out_001.dat" u 1



1. Heisenberg chain [Exact diag. by HΦ]

Sample script for HΦ

HPhi -s stan_hphi.in

```
L               = 4
model           = "Spin"
lattice         = "chain"
method          = "CG"
J               = 1.0
2Sz             = 0
exct            = 4
```

exct: # of excited states

For small, system sizes, you can do full diagonaliztion

```
L               = 4
model           = "Spin"
lattice         = "chain"
method          = "fulldiag"
J               = 1.0
2Sz             = 0
```

1. Heisenberg chain

$$H = \sum_{\langle i,j \rangle} \vec{S}_i \cdot \vec{S}_j$$

`mVMC-tutorial/HandsOn/2022_1128/Samples/1D_Heisenberg`

Step 1: Optimization [stan_opt.in]

output/zvo_out_001.dat

1	2	3	4	5	6
Re[<H>]	Im[<H>]	<H^2>	[<H^2>-<H>^{2}]/<H>^{2}	<S^z>	<(S^z)^2>

1st row: Energy = Re[<H>]
4th row: Variance = [<H^2>-<H>^{2}]/<H>^{2}

By seeing energy, you can check the convergence of optimization.

Note that if the variational wave function becomes an eigenstate, variance=0.
However, VMC is *not* exact method, variance becomes finite in most cases.
Exceptions: small system sizes, non-interacting systems

1. Heisenberg chain

$$H = \sum_{\langle i,j \rangle} \vec{S}_i \cdot \vec{S}_j$$

mVMC-tutorial/HandsOn/2022_1128/Samples/1D_Heisenberg

Step 1: Calc. of 1- and 2-body Green functions for optimized wave functions [stan_aft.in]

stan_aft.in

```
L              = 4
Lsub           = 2
lattice        = "chain"
model          = "Spin"
J              = 1.0
2Sz            = 0
NVMCSample     = 200
NSROptItrStep  = 600
NMPTrans       = 1
NSPStot        = 0
NVMCCalMode    = 1   # 1=calculation of physical quantities, 0=optimization
NDataIdxStart  = 0   #  the first index
NDataQtySmp    = 5   #  number of bins
```

cp ./output/zqp_opt.dat .
mv output opt

vmcdry ./stan_aft.in
cp green1 greenone.def
cp green2 greentwo.def
vmc ./namelist.def ./zqp_opt.dat

NB: output is overwritten!

zqp_opt.dat = optimized wave functions !

1. Heisenberg chain

$$H = \sum_{\langle i,j \rangle} \vec{S}_i \cdot \vec{S}_j$$

`mVMC-tutorial/HandsOn/2022_1128/Samples/1D_Heisenberg`

Step 1: Calc. of 1- and 2-body Green functions for optimized wave functions [stan_aft.in]

ls -l ./output

In output

zvo_out_000.dat - zvo_out_004.dat [energy, variance …]
zvo_cisajs_000.dat - zvo_cisajs_004.dat [1-body Green functions]
zvo_cisajscktalt_000.dat -zvo_cisajscktalt_004.dat [2-body Green functions]

From these files, you can calculate average values and statistical errors of physical quantities such as energy, charge/spin structure factors, etc…

1. Heisenberg chain

$$H = \sum_{\langle i,j \rangle} \vec{S}_i \cdot \vec{S}_j$$

`mVMC-tutorial/HandsOn/2022_1128/Samples/1D_Heisenberg`

Step 1: Calc. of 1- and 2-body Green functions for optimized wave functions [stan_aft.in]

mv output aft

python3 VMClocal.py input.toml
python3 VMCcor.py input.toml

VMClocal.py → Calculation of energies and local charge/spin density
VMCcor.py → Calculation of spin/charge structure factors, and spin correlations

1. Heisenberg chain

$$H = \sum_{\langle i,j \rangle} \vec{S}_i \cdot \vec{S}_j$$

`mVMC-tutorial/HandsOn/2022_1128/Samples/1D_Heisenberg`

Step 1: Calc. of 1- and 2-body Green functions for optimized wave functions [stan_aft.in]

cat Ene.dat

```
# Ene err_Ene Ene/(All_site) err_Ene/(All_site)
-2.000000 0.000000 -0.500000 0.000000
```

mVMC generates exact ground state for L=4

$$\mathrm{occ} = \frac{1}{N_\mathrm{s}} \sum_i (n_{i\uparrow} + n_{i\downarrow})$$

cat occ.dat

```
# occ err_occ AF err_AF
1.000000 0.000000 0.032000 0.038425
```

$$\mathrm{AF} = \frac{1}{N_\mathrm{s}} \sum_i (n_{i\uparrow} - n_{i\downarrow}) e^{i\pi r_i}$$

Averaged charge density occ=1. Antiferromagnetic spin moment is zero AF=0.03(4) with in error bar.

1. Heisenberg chain

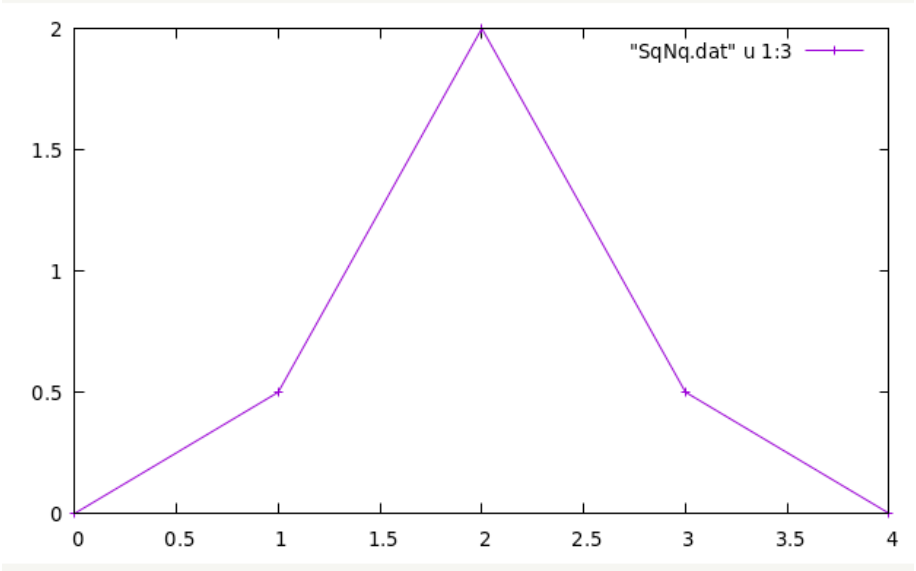
$$H = \sum_{\langle i,j \rangle} \vec{S}_i \cdot \vec{S}_j$$

`mVMC-tutorial/HandsOn/2022_1128/Samples/1D_Heisenberg`

Step 1: Calc. of 1- and 2-body Green functions for optimized wave functions [stan_aft.in]

gnuplot
p "SqNq.dat" u 1:3 w lp

$$S(q) = \frac{1}{N_{\mathrm{s}}} \sum_{i,j} \vec{S}_i \cdot \vec{S}_j e^{iqr_i}$$



(kx/(2π))*Lx

NB: In some cases, pre-factor of S(q) is $1/(3N_s)$.

If you want to check error bars,

gnuplot
p "SqNq.dat" u 1:3:4 w err

1. Heisenberg chain

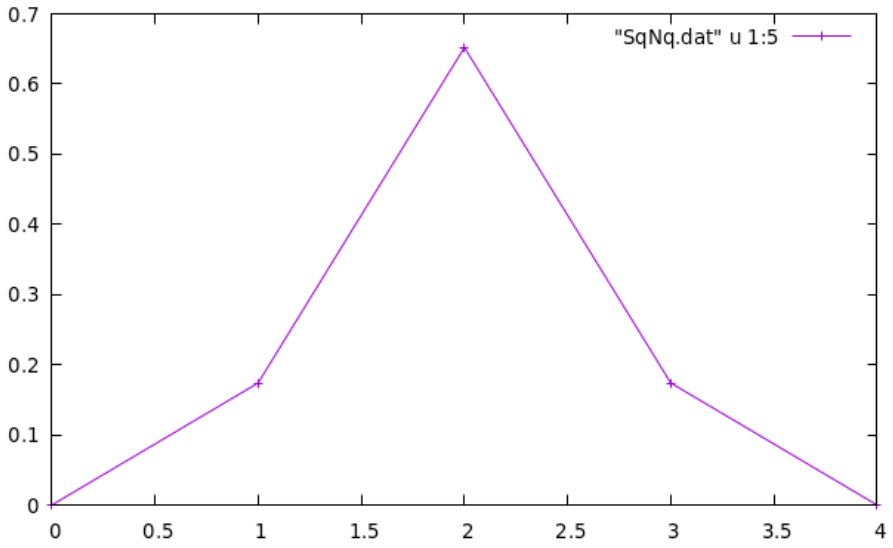
$$H = \sum_{\langle i,j \rangle} \vec{S}_i \cdot \vec{S}_j$$

`mVMC-tutorial/HandsOn/2022_1128/Samples/1D_Heisenberg`

Step 1: Calc. of 1- and 2-body Green functions for optimized wave functions [stan_aft.in]

gnuplot
p "SqNq.dat" u 1:5 w lp

$$S^z(q) = \frac{1}{N_s} \sum_{i,j} S_i^z \cdot S_j^z e^{iqr_i}, S_i^z = (n_{i\uparrow} - n_{i\downarrow})/2$$



(kx/(2π))*Lx

NB:$S^z(q)=S(q)/3$ is satisfied if SU(2) symmetry is preserved.

If you want to check error bars,

gnuplot
p "SqNq.dat" u 1:5:6 w err

1. Heisenberg chain

$$H = \sum_{\langle i,j \rangle} \vec{S}_i \cdot \vec{S}_j$$

Let's change system sizes!
Edit "Lx" in input.toml

```
cp -r 1D_Heisenberg L8_1D_Heisenberg
cd ./L8_1D_Heisenberg

 python3 MakeInput.py input.toml
 vmcdry stan_opt.in
 vmc namelist.def
 cp output/zqp_opt.dat ./
 mv output opt

 vmcdry stan_aft.in
 cp green1 greenone.def
 cp green2 greenone.def
 vmcdry stan_aft.in
 vmc namelist.def ./zqp_opt.dat
 mv output aft

 python3 VMClocal.py input.toml
 python3 VMCcor.py   input.toml
```

For L =8 sites

```
[lattice]
Lx      = 8
Ly      = 1
Lz      = 1
orb_num = 1
model_type = "Spin"
[mVMC]
sub_x   = 2
sub_y   = 1
sub_z   = 1
[mVMC_aft]
modpara   = "modpara.def"
directory = "aft"
```

1. Heisenberg chain (references)

These energies are obtained by exact diagonalization (HΦ)

```
L=6:
 0 -2.8027756377
 1 -2.1180339887
 2 -1.5000000000
 3 -1.2807764064
 4 -1.2807764064
 5 -1.0000000000
 6 -1.0000000000
 7 -0.5000000000
```

```
L=8:
 0 -3.6510934089
 1 -3.1284190638
 2 -2.6996281483
 3 -2.4587385089
 4 -2.4587385089
 5 -2.1451483739
 6 -2.1451483739
 7 -1.8546376797
```

```
L=10:
 0 -4.5154463545
 1 -4.0922073467
 2 -3.7705974354
 3 -3.5432793743
 4 -3.5432793743
 5 -3.2461649167
 6 -3.2461649167
 7 -2.9759318691
```

Check accuracy of mVMC method !

If you change
NSPStot=0 → NSPStot=1 in stan_opt.in & stan_aft.in
you can generate S=1 state.
Please try this! How about S=2, S=3 …?

Scripts X.sh & Clean.sh

sh ./X.sh → Performing all calculations
sh ./Clean.sh → Delete all generated files (Initialization)

After editing input.toml, by executing "sh X.sh", you can do optimization, calculation of Green functions, and post process.

```
#[s] definitions of executions
MPI=" "
VMC="vmc" #MAL
VMCDRY="vmcdry" #MAL
#[e] definitions of executions

python3 MakeInput.py input.toml

#[s] opt
  ${VMCDRY} ./stan_opt.in
  ${MPI} ${VMC} namelist.def
  cp ./output/zqp_opt.dat .
  mv output opt
#[e] opt

#[s] aft
  ${VMCDRY} ./stan_aft.in
  cp green1 greenone.def
  cp green2 greentwo.def
  ${MPI} ${VMC} namelist.def ./zqp_opt.dat
  mv output aft
#[e] aft

#[s] post process
  python3 VMClocal.py  input.toml
  python3 VMCcor.py    input.toml
#[e] post process
```

1. Hubbard chain

mVMC-tutorial/HandsOn/2022_1128/Samples/1D_Hubbard

$$H = -t \sum_{\langle i,j \rangle} (c^{\dagger}_{i\sigma} c_{j\sigma} + \text{h.c.}) + U \sum_{i} n_{i\uparrow} n_{i\downarrow}$$

```
cp -r 1D_Hubbard L4_1D_Hubbard
cd ./L4_1D_Hubbard

 vmcdry stan_opt.in
 vmc namelist.def
 cp output/zqp_opt.dat ./
 mv output opt

 vmcdry stan_aft.in
 cp green1 greenone.def
 cp green2 greentwo.def

 vmc namelist.def ./zqp_opt.dat
 mv output aft

 python3 VMClocal.py input.toml
 python3 VMCcor.py   input.toml
```

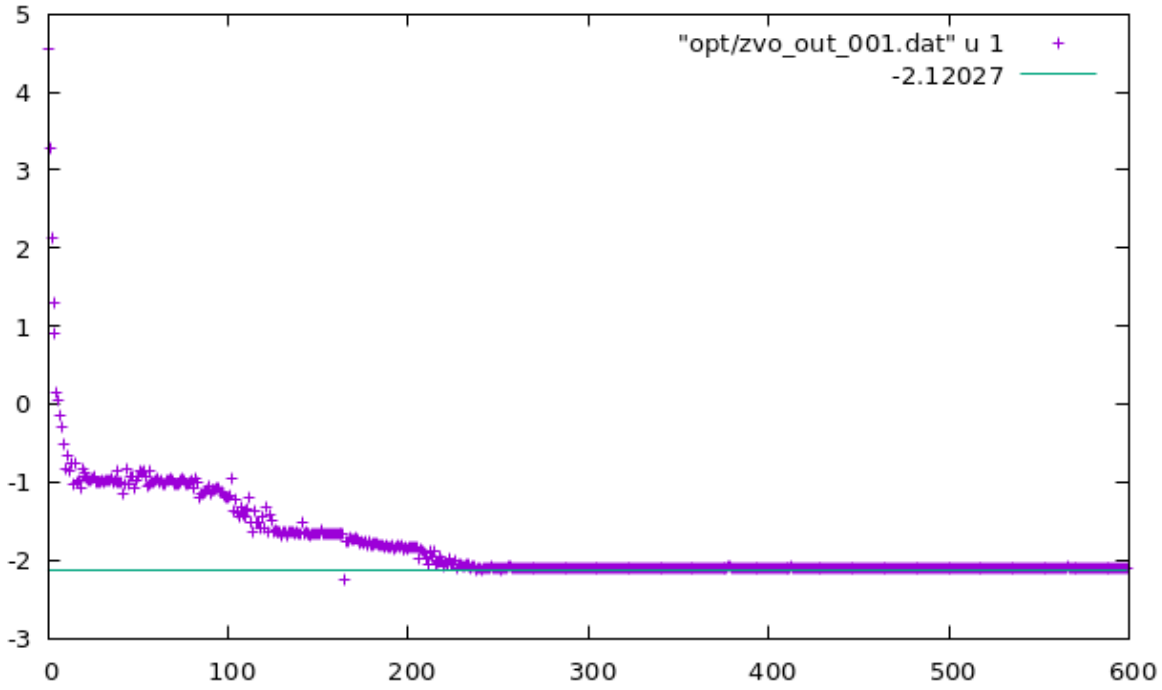
```
L              = 4
Lsub           = 2
lattice        = "chain"
model          = "Hubbard"
t              = 1.0
U              = 4.0
ncond          = 4
2Sz            = 0
NVMCSample     = 200
NSROptItrStep  = 600
NMPTrans       = 1
NSPStot        = 0
```

NB:
ncond = L → half filling = one electron/site

1. Hubbard chain

$$H = -t \sum_{\langle i,j \rangle} (c^{\dagger}_{i\sigma} c_{j\sigma} + \text{h.c.}) + U \sum_{i} n_{i\uparrow} n_{i\downarrow}$$

plot "./opt/zvo_out_001.dat" u 1




```
Energy by HΦ
 0 -2.1027484835
 1 -1.8064238518
 2 -1.0681403934
 3 -0.8284271247
 4 -0.8284271247
 5 0.0000000000
 6 0.5814492811
 7 2.0000000000
```

1. Hubbard chain [Exact diag. by HΦ]

$$H = -t \sum_{\langle i,j \rangle} (c^{\dagger}_{i\sigma} c_{j\sigma} + \text{h.c.}) + U \sum_{i} n_{i\uparrow} n_{i\downarrow}$$

Sample script for HΦ **HPhi -s stan_hphi.in**

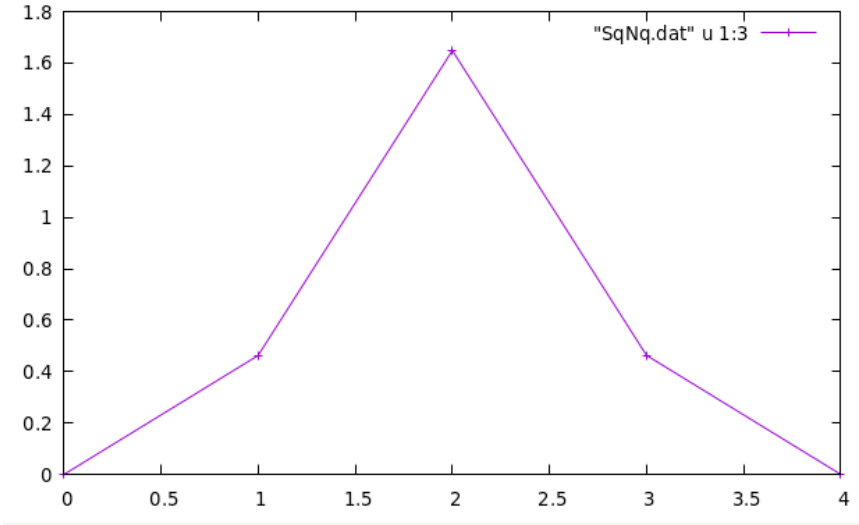
```
L                = 4
model            = "Hubbard"
lattice          = "chain"
method           = "CG"
U                = 4.0
t                = 1.0
2Sz              = 0
nelec            = 4
exct             = 8
```

1. Hubbard chain [Exact diag. by HΦ]

$$H = -t \sum_{\langle i,j \rangle} (c^{\dagger}_{i\sigma} c_{j\sigma} + \text{h.c.}) + U \sum_{i} n_{i\uparrow} n_{i\downarrow}$$

p "SqNq.dat" u 1:3 w lp

$$S(q) = \frac{1}{N_{\text{s}}} \sum_{i,j} \vec{S}_i \cdot \vec{S}_j e^{iqr_i}$$

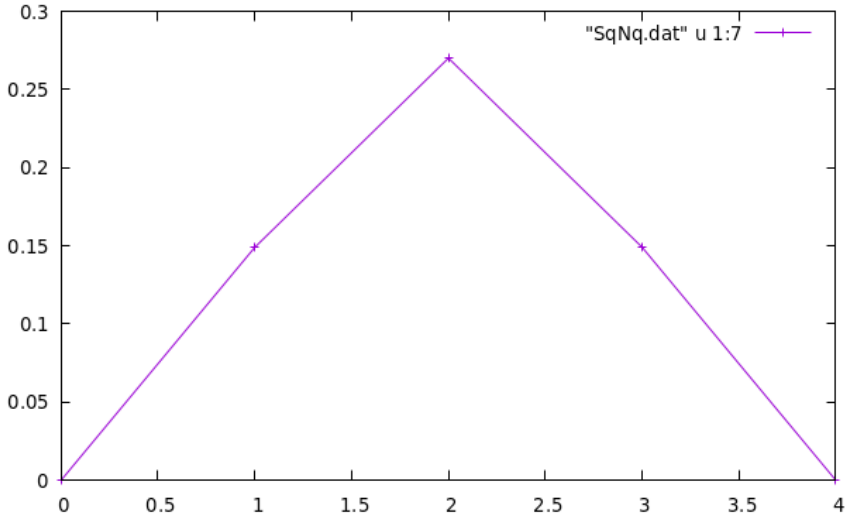


(kx/(2π))*Lx

p "SqNq.dat" u 1:7 w lp

$$N(q) = \frac{1}{N_{\text{s}}} \sum_{i,j} \bar{n}_i \cdot \bar{n}_j e^{iqr_i},$$

$$\bar{n}_i = (n_{i\uparrow} + n_{i\downarrow}) - \langle (n_{i\uparrow} + n_{i\downarrow}) \rangle$$



(kx/(2π))*Lx

1. Hubbard chain

Let's change system sizes!
Edit "Lx" in input.toml

python3 MakeInput.py input.toml

$$H = -t \sum_{\langle i,j \rangle} (c^{\dagger}_{i\sigma} c_{j\sigma} + \text{h.c.}) + U \sum_{i} n_{i\uparrow} n_{i\downarrow}$$

```
cp -r 1D_Hubbard L6_1D_Hubbard
cd ./L6_1D_Hubbard

 vmcdry stan_opt.in
 vmc namelist.def
 cp output/zqp_opt.dat ./
 mv output opt

 vmcdry stan_aft.in
 cp green1 greenone.def
 cp green2 greentwo.def

 vmc namelist.def ./zqp_opt.dat
 mv output aft

 python3 VMClocal.py input.toml
 python3 VMCcor.py   input.toml
```

```
L             = 6
Lsub          = 2
lattice       = "chain"
model         = "Hubbard"
t             = 1.0
U             = 4.0
ncond         = 4
2Sz           = 0
NVMCSample    = 200
NSROptItrStep = 600
NMPTrans      = 1
NSPStot       = 0
```

NB:
ncond = L → half filling = one electron/site

1. Hubbard chain (references)

These energies are obtained by exact diagonalization (HΦ)

L=6:

0	−3.668706178872957 1
1	−2.898381474036730 4
2	−2.516376873116143 1
3	−2.422911263847928 9
4	−2.422911263847929 3
5	−2.092753829496921 0
6	−2.092753829496921 0
7	−1.769024823288434 5

L=8:

0	−4.603526299892002
1	−4.299992758433059 9
2	−4.010153957644034 2
3	−3.705764239483940 5
4	−3.705764239483940 5
5	−3.496356310215205 1
6	−3.496356310215204 2
7	−3.244557098464969 4

Check accuracy of mVMC method !

Advanced:
Perform the total momentum projections for L=6, 8
by specifying NMPTrans = 2 in stan_opt.in & stan_aft.in

2. Heisenberg & Hubbard on the square lattice

- **Procedure is basically the same as that of Heisenberg and Hubbard chain. Please note that the elapsed time becomes longer !**

`mVMC-tutorial/HandsOn/2022_1128/Samples/2D_Heisenberg`

stan_opt.in

```
W             = 4
Wsub          = 2
L             = 4
Lsub          = 2
lattice       = "square"
model         = "Spin"
J             = 1.0
2Sz           = 0
NVMCSample    = 200
NSROptItrStep = 600
NMPTrans      = 1
NSPStot       = 0
```

`mVMC-tutorial/HandsOn/2022_1128/Samples/2D_Hubbard`

stan_opt.in

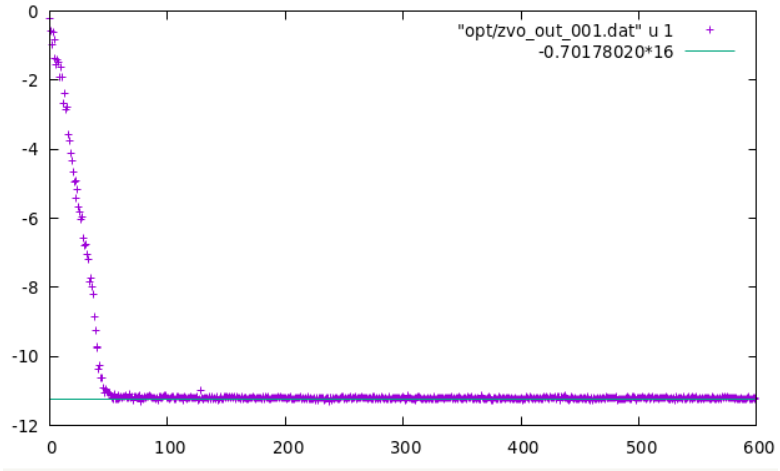
```
W             = 4
Wsub          = 2
L             = 4
Lsub          = 2
lattice       = "square"
model         = "Hubbard"
t             = 1.0
U             = 4.0
ncond         = 16
2Sz           = 0
NVMCSample    = 200
NSROptItrStep = 600
NMPTrans      = 1
NSPStot       = 0
```

2. Heisenberg & Hubbard on the square lattice

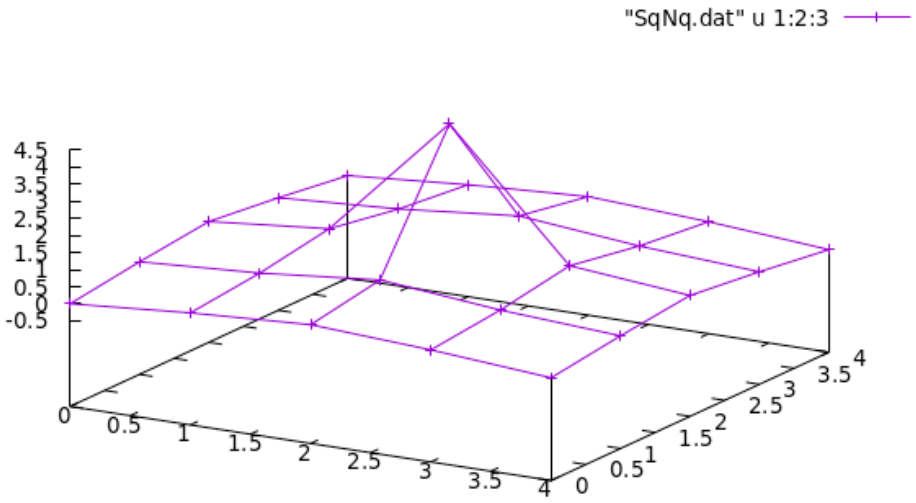
- **Procedure is basically the same as that of Heisenberg and Hubbard chain. Please note that the elapsed time becomes longer !**

4by4 Heisenberg model, J=1

plot "./op/zvo_out_001.dat" u 1



sp "SqNq.dat" u 1:2:3 w lp

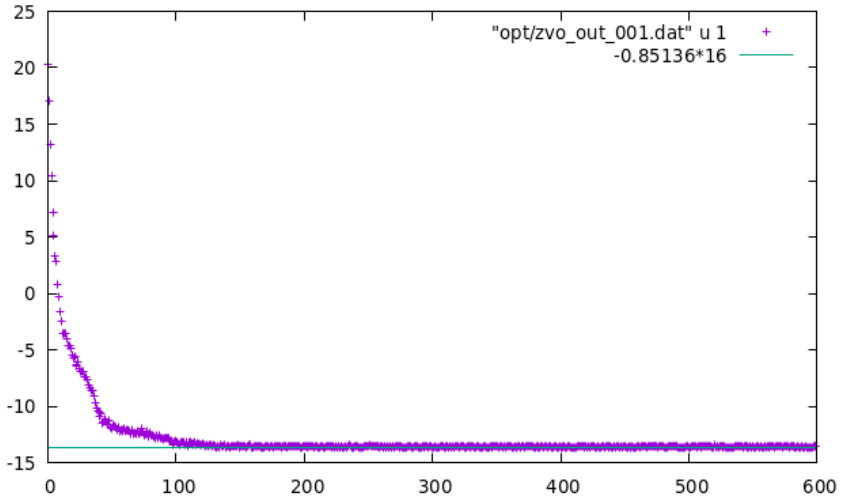


2. Heisenberg & Hubbard on the square lattice

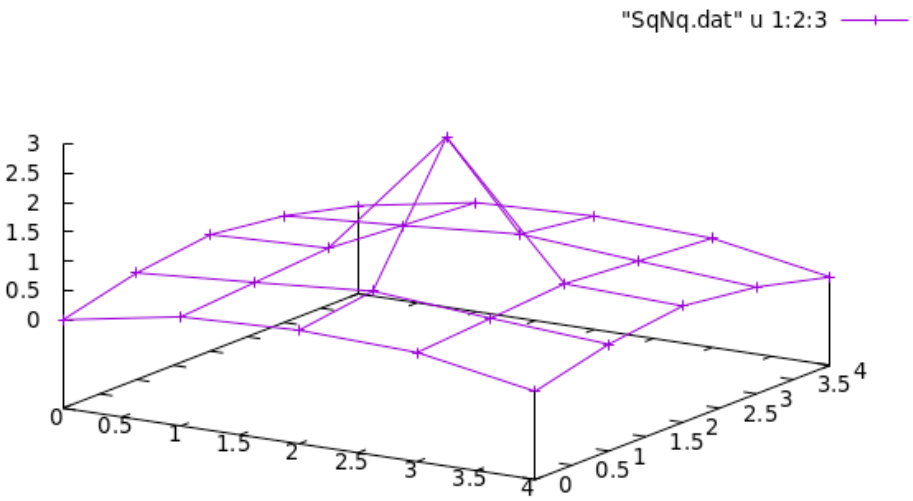
- **Procedure is basically the same as that of Heisenberg and Hubbard chain. Please note that the elapsed time becomes longer !**

4by4 Hubbard model, t=1,U=4, ncond=16

plot "./opt/zvo_out_001.dat" u 1



sp "SqNq.dat" u 1:2:3 w lp



2. Heisenberg & Hubbard on the square lattice

Reference data for Heisenberg model

T. Misawa et al., Comp. Phys. Comm. 235, 447 (2019).

Table 4

Comparisons with exact diagonalization for 4×4 and 6×6 Heisenberg model with $J = 1$. We note $\boldsymbol{k}_{\text{peak}} = (\pi, \pi)$. The relative errors η become $10^{-6}\%$ for $L = 4$ and $10^{-3}\%$ for $L = 6$, respectively. The definitions of the spin correlations in the Lanczos method and N_p are same as those of Table 3.

$(N_s = 4 \times 4)$	E/N_s	S_{nn}	S_{nnn}	$\tilde{S}(\boldsymbol{k}_{\text{peak}})$	N_p
ED	−0.70178020	−0.35089010	0.21376	0.09217	–
mVMC(2 × 2)	−0.701765(2)	−0.350883(1)	0.2136(1)	0.09216(3)	64
mVMC(2 × 2)+Lanczos	−0.701780(1)	−0.3517(5)	0.214(1)	0.0924(2)	64
mVMC(4 × 4)	−0.70178015(8)	−0.35089007(4)	0.2139(4)	0.0922(1)	256
$(N_s = 6 \times 6)$	E/N_s	S_{nn}	S_{nnn}	$\tilde{S}(\boldsymbol{k}_{\text{peak}})$	N_p
ED	−0.678872	−0.33943607	0.207402499	0.069945	–
mVMC(2 × 2)	−0.67846(1)	−0.33923(1)	0.20742(3)	0.07021(2)	144
mVMC(2 × 2)+Lanczos	−0.678840(4)	−0.339(1)	0.207(1)	0.0698(3)	144
mVMC(6 × 6)	−0.678865(1)	−0.3394326(4)	0.20735(4)	0.06993(3)	1296
mVMC(6 × 6)+Lanczos	−0.678871(1)	−0.3391(5)	0.2071(6)	0.0699(2)	1296

Note: Si*Sj is output in “Sij.dat”

$$\tilde{S}(q) = \frac{S(q)}{3N_{\text{s}}^2} = \frac{1}{3N_{\text{s}}^2} \sum_{i,j} \vec{S}_i \cdot \vec{S}_j e^{iqr_i}$$

2. Heisenberg & Hubbard on the square lattice

Reference data for Hubbard model

T. Misawa et al., Comp. Phys. Comm. 235, 447 (2019).

Table 3

Comparisons with exact diagonalization (ED) for 4×4 Hubbard model with $U = 4$ and $t = 1$ at half filling. ED is done by using $\mathcal{H}\Phi$ [7,57]. mVMC($2 \times 2/4 \times 4$) means f_{ij} has the $2 \times 2/4 \times 4$ sublattice structure. N_p is the size of S matrix, which is the number of variational parameters and $\boldsymbol{k}_{\text{peak}} = (\pi, \pi)$. Error bars are denoted by the parentheses in the last digit. Lanczos means that the first-step Lanczos calculations on top of the mVMC calculations. In the Lanczos calculations, to reduce the numerical cost, we calculate the diagonal spin correlations such as $S^z_{\text{nn}} = 3/4N_s \sum_{i,\mu} \langle S^z_{\boldsymbol{r}_i} \cdot S^z_{\boldsymbol{r}_i+\boldsymbol{e}_\mu} \rangle$ and $\tilde{S}^z(\boldsymbol{k}) = S^z(\boldsymbol{k})/N_s$, which are equivalent to S_{nn} and $\tilde{S}(\boldsymbol{k})$ when the spin-rotational symmetry is preserved.

	E/N_s	D	S_{nn}	$\tilde{S}(\boldsymbol{k}_{\text{peak}})$	N_p
ED	−0.85136	0.11512	−0.2063	0.05699	–
mVMC(2×2)	−0.84985(3)	0.1155(1)	−0.2057(2)	0.05762(4)	74
mVMC(2×2)+Lanczos	−0.85100(2)	0.1156(1)	−0.2054(8)	0.05736(2)	74
mVMC(4×4)	−0.85070(2)	0.1151(1)	−0.2065(1)	0.05737(2)	266
mVMC(4×4)+Lanczos	−0.85122(1)	0.1151(1)	−0.2072(4)	0.0576(1)	266

2. Heisenberg & Hubbard on the square lattice

Sij.dat

# i	j	tot_S	err_tot_S	Sxy	err_Sxy	Sz	err_Sz
0	0	0.75000000	0.00000000	0.50000000	0.00000000	0.25000000	0.00000000
0	1	-0.34746908	0.01192859	-0.22546908	0.00706604	-0.12200000	0.01287925
0	2	0.21767184	0.00700917	0.14167184	0.01234442	0.07600000	0.01433091

tot_S : $\vec{S}_i \cdot \vec{S}_j = S_i^x S_j^x + S_i^y S_j^y + S_i^z S_j^z$

Sxy : $S_i^x S_j^x + S_i^y S_j^y$

Szz : $S_i^z S_j^z$

$$S_i^x = \frac{1}{2}(S_i^+ + S_i^-) = \frac{1}{2}(c_{i\uparrow}^\dagger c_{i\downarrow} + c_{i\downarrow}^\dagger c_{i\uparrow})$$

$$S_i^y = \frac{i}{2}(-S_i^+ + S_i^-) = \frac{i}{2}(-c_{i\uparrow}^\dagger c_{i\downarrow} + c_{i\downarrow}^\dagger c_{i\uparrow})$$

$$S_i^z = \frac{1}{2}(n_{i\uparrow} - n_{i\downarrow}) = \frac{1}{2}(c_{i\uparrow}^\dagger c_{i\uparrow} - c_{i\downarrow}^\dagger c_{i\downarrow})$$

In mVMC, spin operators are implemented by Abrikosov fermion representation.

$$S_i^x S_j^x + S_i^y S_j^y = \frac{1}{2}(S_i^+ S_j^- + S_i^- S_j^+) = \frac{1}{2}(c_{i\uparrow}^\dagger c_{i\downarrow} c_{j\downarrow}^\dagger c_{j\uparrow} + c_{i\downarrow}^\dagger c_{i\uparrow} c_{j\uparrow}^\dagger c_{j\downarrow})$$

$$= -\frac{1}{2}(c_{i\uparrow}^\dagger c_{j\uparrow} c_{j\downarrow}^\dagger c_{i\downarrow} + c_{i\downarrow}^\dagger c_{j\downarrow} c_{j\uparrow}^\dagger c_{i\uparrow}) + \frac{1}{2}\delta_{ij}(c_{i\uparrow}^\dagger c_{j\uparrow} + c_{i\downarrow}^\dagger c_{j\downarrow})$$

$$S_i^z S_j^z = \frac{1}{4}(n_{i\uparrow} n_{j\uparrow} - n_{i\uparrow} n_{j\downarrow} - n_{i\uparrow} n_{j\downarrow} + n_{i\downarrow} n_{i\downarrow})$$

3. Initial states from UHF calculations

Generating initial states from unrestricted Hartree-Fock calculations.
Execution file for UHF calculations (UHF) exists in
`/usr/share/mvmc/tool/UHF`

mVMC-tutorial/HandsOn/2022_1128/Samples/IniUHF

Ex. 2D Hubbard, 4by4, t=1,U=4,

vmcdry stan_opt.in

python3 MakeIni.py input.toml
echo " Initial initial.def" >> namelist.def
cp /usr/share/mvmc/tool/UHF .

./UHF namelist.def

```
==================
initial 32
==================
==================
==================
 0 0 0 0 0.100000 0.000000
 0 1 0 1 -0.100000 0.000000
 1 0 1 0 -0.100000 0.000000
 1 1 1 1 0.100000 0.000000
 2 0 2 0 0.100000 0.000000
 2 1 2 1 -0.100000 0.000000
```

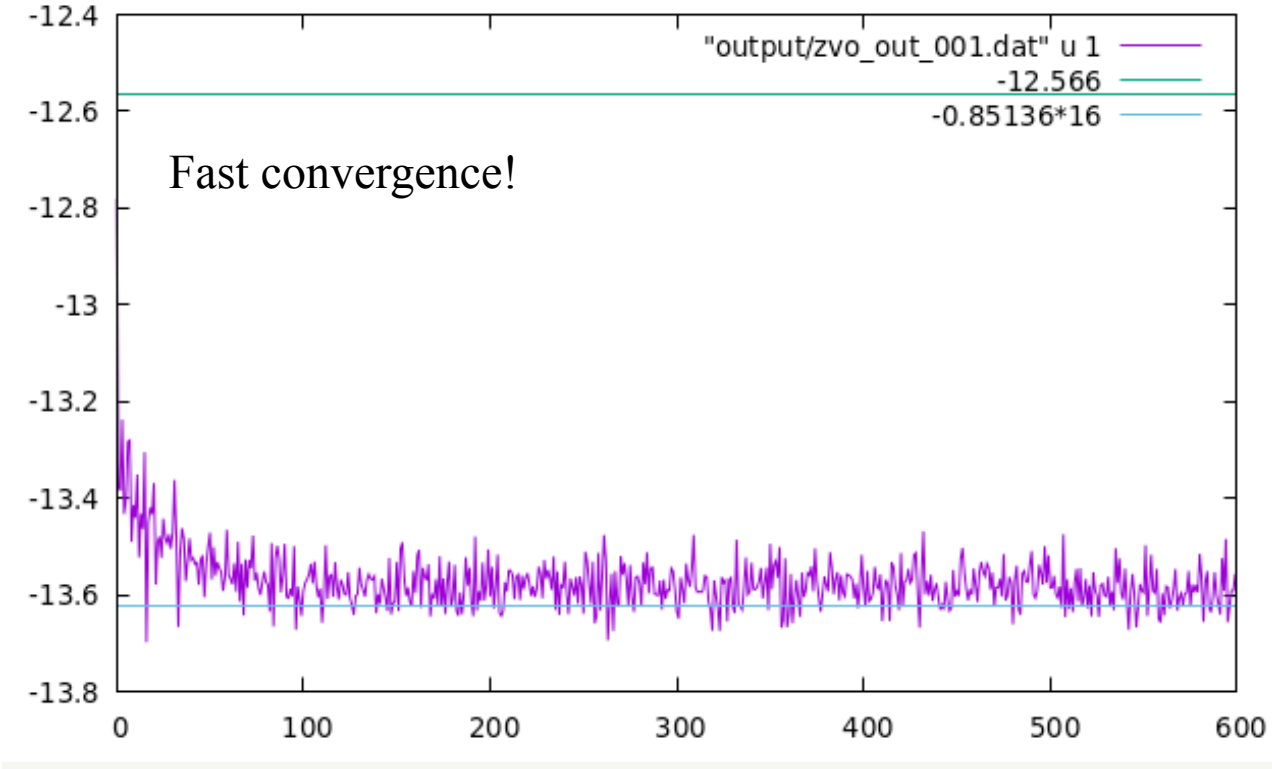
```
$ cat zvo_result.dat
 energy -12.5665543126
 num    16.0000000000
```

zvo_result.dat: energy by UHF

Tips: Proper initial Green functions (Weiss fields) are necessary
MakeIni.py →Generating Green functions for (π, π) magnetic order
Add "Initial initial.def" in namelist.def

3. Initial states from UHF calculations

vmcdry stan_opt.in
echo " InOrbital zqp_APOrbital_opt.dat" >> namelist.def
vmc namelist.def



zqp_APOrbital_opt.dat
f_{ij} generated by UHF

$$|\phi_{\text{AP-Pf}}\rangle = \left(\sum_{i,j=0}^{N_s-1} f_{ij} c_{i\uparrow}^{\dagger} c_{j\downarrow}^{\dagger}\right)^{N_e/2} |0\rangle.$$

$$|\phi_{\text{SL}}\rangle = \left(\prod_{n=1}^{N_e} \psi_n^{\dagger}\right)|0\rangle, \quad \psi_n^{\dagger} = \sum_{l=1}^{2N_s} \Phi_{ln} c_l^{\dagger},$$

$$f_{ij} = \sum_{n=1}^{N_e/2} \Phi_{in\uparrow} \Phi_{jn\downarrow}.$$

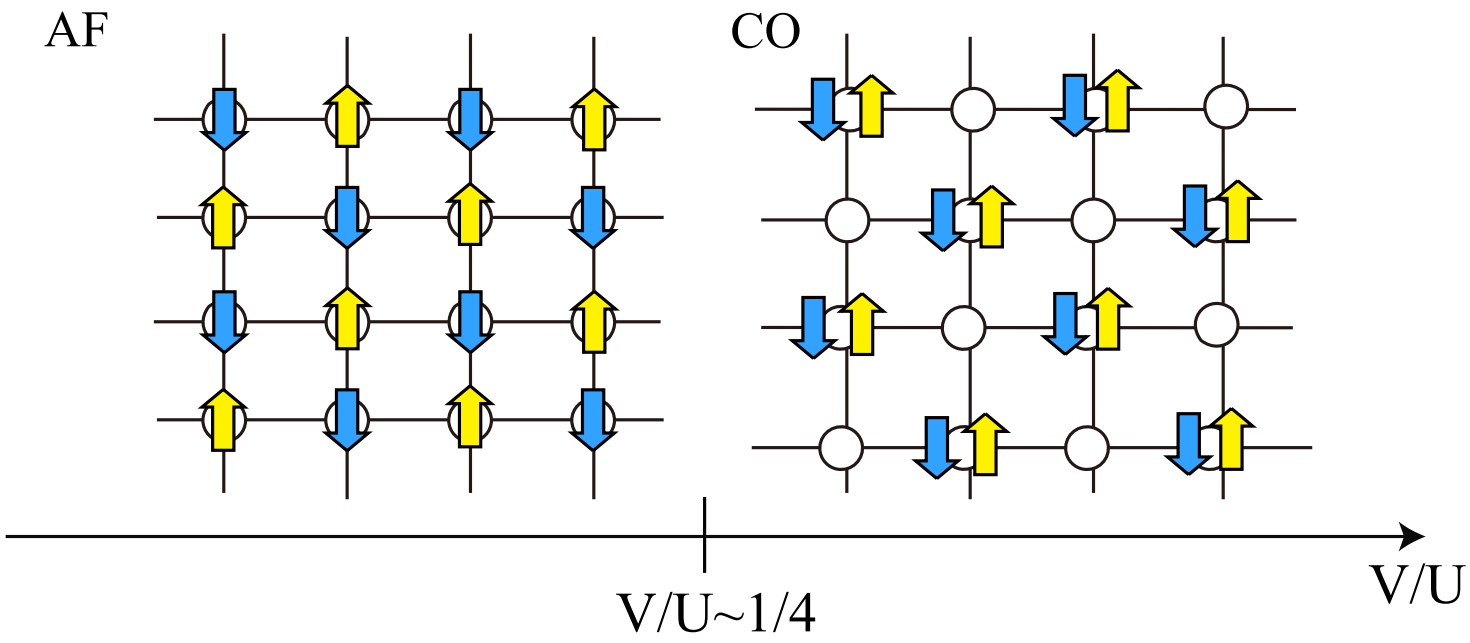
NB: $E_{UHF} > E_{mVMC}$ should be satisfied. If not, something is wrong.

Advanced exercise
(Let's generate quantum phases by mVMC)

1. Hubbard + V → Charge order
2. Heisenberg+J2 → stripe magnetic order
3. Attractive Hubbard → superconductivity
4. 1D Kondo lattice → Kondo insulator
5. Data used in researches
[Data repository in ISSP]

1. Introducing off-site Coulomb interaction V

$$H_V = V \sum_{\langle i,j \rangle} n_i n_j$$



Check whether the ground state is charge-ordered phase by plotting SqNq.dat.

1. Introducing off-site Coulomb interaction V

$$H_V = V \sum_{\langle i,j \rangle} n_i n_j$$

Ex. stan_opt.in

```
W                = 4
Wsub             = 2
L                = 4
Lsub             = 2
lattice          = "square"
model            = "Hubbard"
t                = 1.0
U                = 4.0
V                = 1.5
ncond            = 16
2Sz              = 0
NVMCSample       = 200
NSROptItrStep    = 600
NMPTrans         = 1
NSPStot          = 0
```

How to rewrite input files

1. python3 MakeInput.py input.toml
→ generating stan_opt.in, stan_aft.in

2. Add "V=1.5" in stan_opt.in & stan_aft. in

3. sh ./X.sh

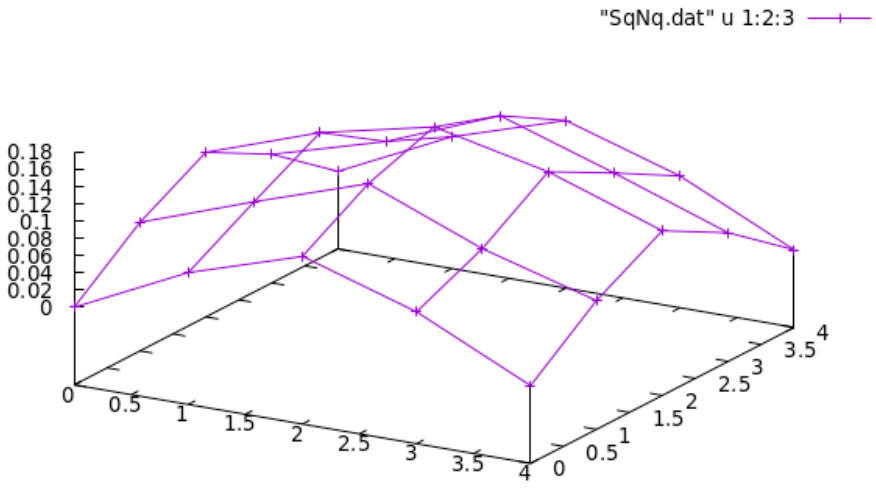
Or, you can directly modify MakeInput.py

1. Introducing off-site Coulomb interaction V

$$H_V = V \sum_{\langle i,j \rangle} n_i n_j$$

sp “SqNq.dat” u 1:2:3 w lp

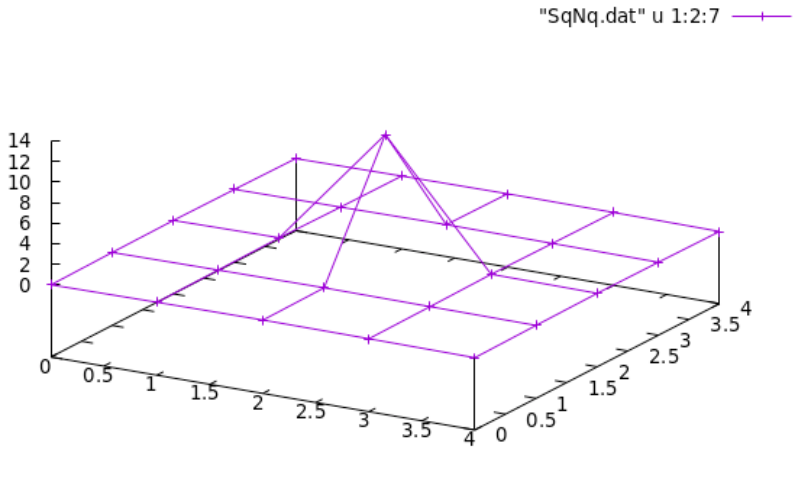
$$S(q) = \frac{1}{N_s} \sum_{i,j} \vec{S}_i \cdot \vec{S}_j e^{iqr_i}$$



sp “SqNq.dat” u 1:2:7 w lp

$$N(q) = \frac{1}{N_s} \sum_{i,j} \bar{n}_i \cdot \bar{n}_j e^{iqr_i},$$

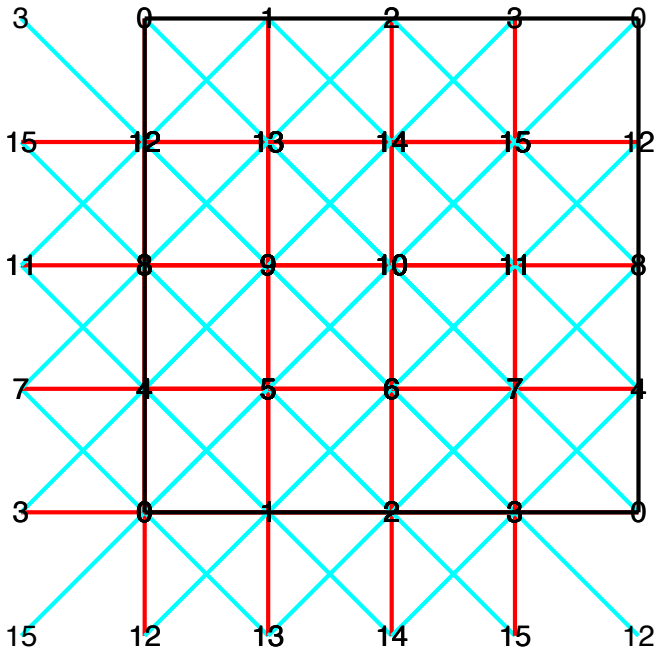
$$\bar{n}_i = (n_{i\uparrow} + n_{i\downarrow}) - \langle (n_{i\uparrow} + n_{i\downarrow}) \rangle$$

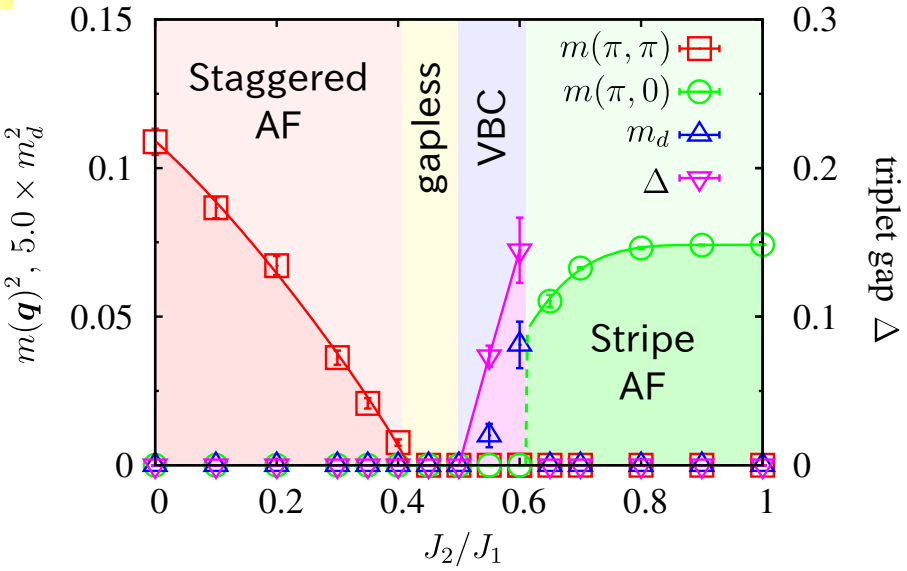


2. J1-J2 Heisenberg model

$$H = J_1 \sum_{\langle i,j \rangle} S_i S_j + J_2 \sum_{\langle\langle i,j \rangle\rangle} S_i S_j$$

nearest J1,next-nearest J2





S. Motira and M. Imada JPSJ (2014).
see also Y. Nomura and M. Imada PRX (2021).

plot by lattice.gp

QSL may appear around J2/J1~0.5

2. J1-J2 Heisenberg model

- Confirmt that stripe magnetic order occurs at J2=1 from SqNq.dat
- What happens around J2=0.5 ?

Ex. stan_opt.in

```
$ cat stan_opt.in
W               = 4
Wsub            = 2
L               = 4
Lsub            = 2
lattice         = "square"
model           = "Spin"
J               = 1.0
J'              = 1.0
2Sz             = 0
NVMCSample      = 200
NSROptItrStep = 600
NMPTrans        = 1
NSPStot         = 0
```

How to rewrite input files

1. python3 MakeInput.py input.toml
→ generating stan_opt.in, stan_aft.in

2. Add "J'=1.0" in stan_opt.in & stan_aft. in

3. sh ./X.sh

Or, you can directly modify MakeInput.py

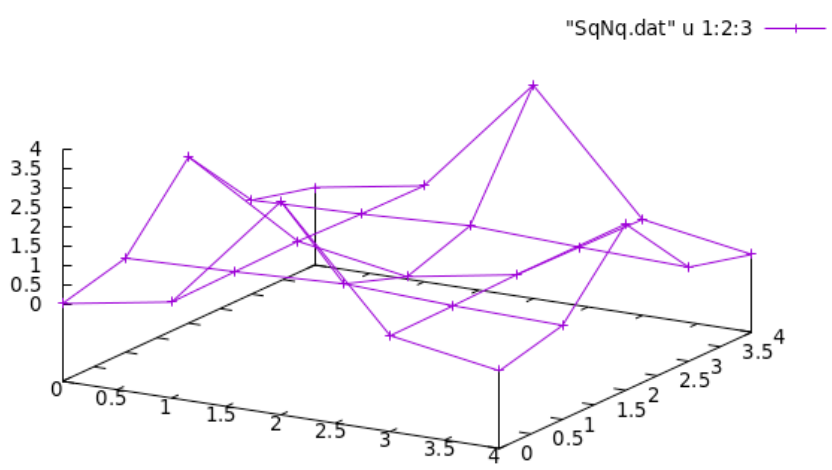
2. J1-J2 Heisenberg model

- Confirm that stripe magnetic order occurs at J2=1 from SqNq.dat
- What happens around J2=0.5 ?

$$S(q) = \frac{1}{N_{\mathrm{s}}} \sum_{i,j} \vec{S}_i \cdot \vec{S}_j e^{iqr_i}$$

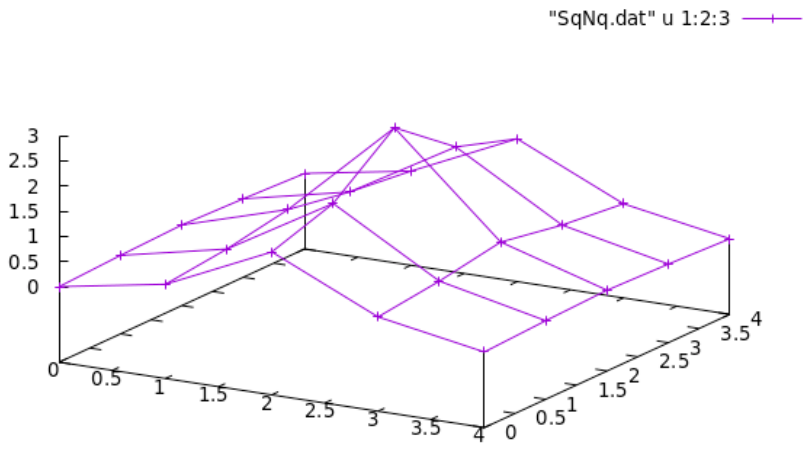
sp “SqNq.dat” u 1:2:3 w lp

J’/J=1



sp “SqNq.dat” u 1:2:3 w lp

J’/J=0.5



2. J1-J2 Heisenberg model

- Cnfirmt that stripe magnetic order occurs at J2=1 from Result_Sq.dat or using fourier tool
- What happens around J2=0.5 ?

Ex. stan_opt.in

```
W                = 4
L                = 4
Wsub             = 2
Lsub             = 2
model            = "Spin"
lattice          = "Tetragonal"
J                = 1.0
J'               = 1.0
2Sz              = 0
NVMCSample       = 50
NSROptItrStep    = 500
NSROptItrSmp     = 100
NMPTrans         = 1
NSPGaussLeg      = 1
```

How to rewrite input files

1. python3 MakeInput.py input.toml
→ generating stan_opt.in, stan_aft.in

2. Add "J'=1.0" in stan_opt.in & stan_aft. in

3. sh ./X.sh

Or, you can directly modify MakeInput.py

3.Attractive Hubbard model

- **Set U as negative values, e.g, U=-4**
- **Calculate 1s SC correlations**

(it is better to perform calculation for doped case)

```
W                = 4
Wsub             = 2
L                = 4
Lsub             = 2
lattice          = "square"
model            = "Hubbard"
t                = 1.0
U                = -4.0
ncond            = 10
2Sz              = 0
NVMCSample       = 200
NSROptItrStep    = 600
NMPTrans         = 1
NSPStot          = 0
```

$$P_{1s}(\vec{r}) = \frac{1}{2N_s}\sum_{i=1}^{N_s}\langle\Delta_{1s}^{\dagger}(\vec{r_i})\Delta_{1s}(\vec{r_i}+\vec{r}) + \Delta_{1s}(\vec{r_i})\Delta_{1s}^{\dagger}(\vec{r_i}+\vec{r})\rangle$$

$$\Delta_{1s}(\vec{r_i}) = \frac{1}{\sqrt{2}}(c_{i\uparrow}c_{i\downarrow} - c_{i\downarrow}c_{i\uparrow}).$$

- Scripts for calculating 1s SC correlating exists at Samples/SC_Correlation

python3 SCGreen.py input.tom # generating SC_1swave

python3 SCGreen.py # result is output in Result_1swave

3.Attractive Hubbard model

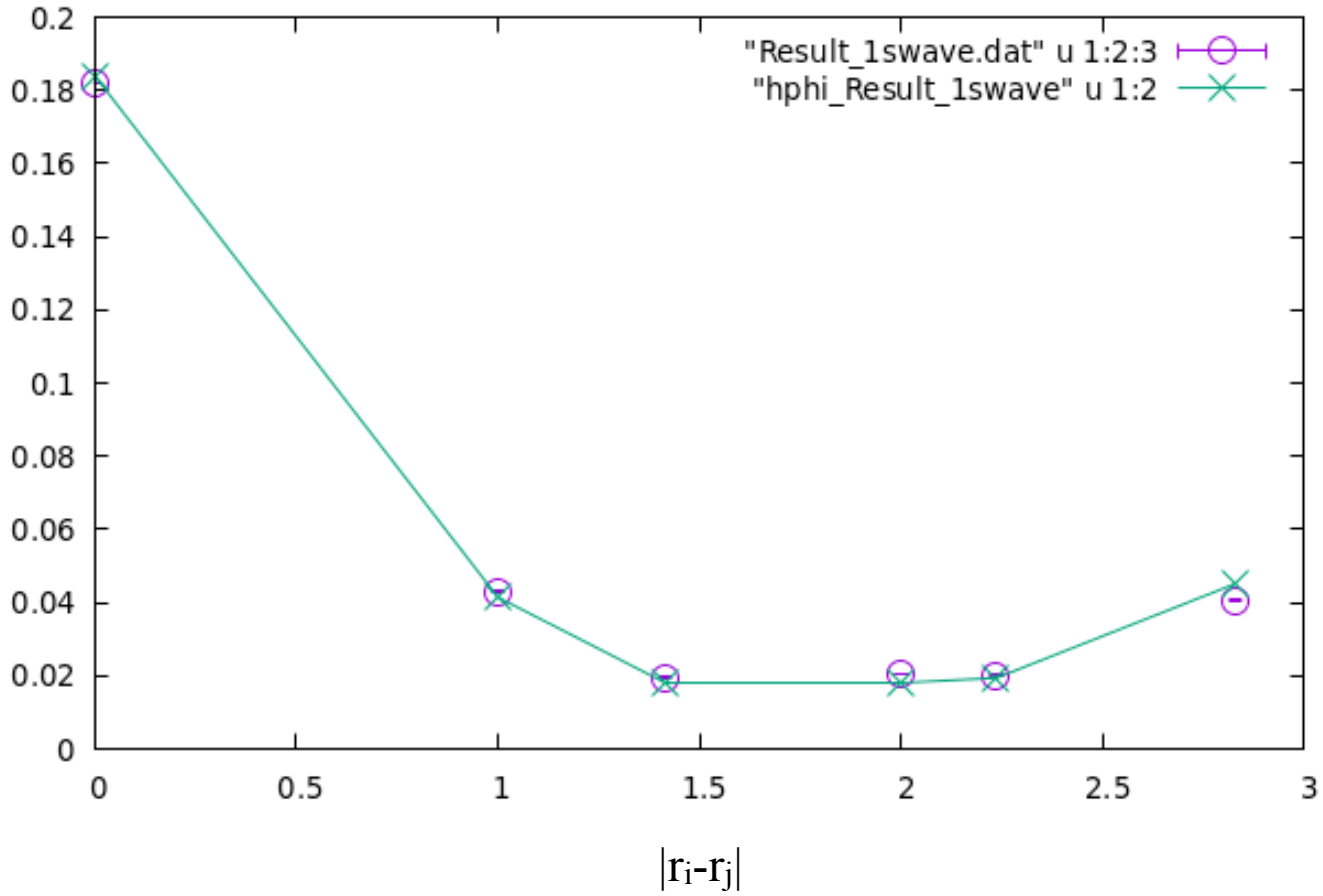
- **Set U as negative values, e.g, U=-4**
- **Calculate 1s SC correlations**

(it is better to perform calculation for doped case)

HPhi:U=-4, ncond=10
hphi_Ene,
hphi_Result_1swave

$\max |P(r_i-r_j)|$

p "Result_1swave.dat" u 1:2:3 w e ps 2 pt 6,
"hpi_Result_1swave.dat" u 1:2 w lp ps 2



3.Attractive Hubbard model

Details of SC correlations:

$$P(\boldsymbol{r}_j-\boldsymbol{r}_i)=\frac{1}{2N_{\mathrm{s}}}\sum_{\boldsymbol{r}_i}\langle\Delta_\alpha^\dagger(\boldsymbol{r}_i)\Delta_\alpha(\boldsymbol{r}_j)+\Delta_\alpha(\boldsymbol{r}_i)\Delta_\alpha^\dagger(\boldsymbol{r}_j)\rangle$$

Details of SC order parameter: $f_\alpha(e)$ is a form factor

$$\Delta_\alpha(\boldsymbol{r}_i)=\frac{1}{\sqrt{2}}\sum_{\boldsymbol{e}}f_\alpha(\boldsymbol{e})(c_{\boldsymbol{r}_i\uparrow}c_{\boldsymbol{r}_i+\boldsymbol{e}\downarrow}-c_{\boldsymbol{r}_i\downarrow}c_{\boldsymbol{r}_i+\boldsymbol{e}\uparrow})$$

$$\Delta_\alpha^\dagger(\boldsymbol{r}_i)=\frac{1}{\sqrt{2}}\sum_{\boldsymbol{e}}f_\alpha(\boldsymbol{e})(c_{\boldsymbol{r}_i+\boldsymbol{e}\downarrow}^\dagger c_{\boldsymbol{r}_i\uparrow}^\dagger-c_{\boldsymbol{r}_i+\boldsymbol{e}\uparrow}^\dagger c_{\boldsymbol{r}_i\downarrow}^\dagger)$$

$$\begin{aligned}\Delta_\alpha^\dagger(\boldsymbol{r}_i)\Delta_\alpha(\boldsymbol{r}_j)=\frac{1}{2}\sum_{\boldsymbol{e},\boldsymbol{e}'}f_\alpha(\boldsymbol{e})f_\alpha(\boldsymbol{e}')\Big(&\\ &+c_{\boldsymbol{r}_i\uparrow}^\dagger c_{\boldsymbol{r}_j\uparrow}c_{\boldsymbol{r}_i+\boldsymbol{e}\downarrow}^\dagger c_{\boldsymbol{r}_j+\boldsymbol{e}'\downarrow}\\ &+c_{\boldsymbol{r}_i\uparrow}^\dagger c_{\boldsymbol{r}_j+\boldsymbol{e}'\uparrow}c_{\boldsymbol{r}_i+\boldsymbol{e}\downarrow}^\dagger c_{\boldsymbol{r}_j\downarrow}\\ &+c_{\boldsymbol{r}_i+\boldsymbol{e}\uparrow}^\dagger c_{\boldsymbol{r}_j+\boldsymbol{e}'\uparrow}c_{\boldsymbol{r}_i\downarrow}^\dagger c_{\boldsymbol{r}_j+\boldsymbol{e}'\downarrow}\\ &+c_{\boldsymbol{r}_i+\boldsymbol{e}\uparrow}^\dagger c_{\boldsymbol{r}_j+\boldsymbol{e}'\uparrow}c_{\boldsymbol{r}_i\downarrow}^\dagger c_{\boldsymbol{r}_j\downarrow}\Big)\end{aligned}$$

$$\begin{aligned}\Delta_\alpha(\boldsymbol{r}_i)\Delta_\alpha^\dagger(\boldsymbol{r}_j)=\frac{1}{2}\sum_{\boldsymbol{e},\boldsymbol{e}'}f_\alpha(\boldsymbol{e})f_\alpha(\boldsymbol{e}')\Big(&\\ &+(\delta_{i,j}-c_{\boldsymbol{r}_j\uparrow}^\dagger c_{\boldsymbol{r}_i\uparrow})(\delta_{i+\boldsymbol{e},j+\boldsymbol{e}'}-c_{\boldsymbol{r}_j+\boldsymbol{e}'\downarrow}^\dagger c_{\boldsymbol{r}_i+\boldsymbol{e}\downarrow})\\ &+(\delta_{i,j+\boldsymbol{e}}-c_{\boldsymbol{r}_j+\boldsymbol{e}'\uparrow}^\dagger c_{\boldsymbol{r}_i\uparrow})(\delta_{i+\boldsymbol{e},j}-c_{\boldsymbol{r}_j\downarrow}^\dagger c_{\boldsymbol{r}_i+\boldsymbol{e}\downarrow})\\ &+(\delta_{i+\boldsymbol{e},j}-c_{\boldsymbol{r}_j\uparrow}^\dagger c_{\boldsymbol{r}_i+\boldsymbol{e}\uparrow})(\delta_{i,j+\boldsymbol{e}'}-c_{\boldsymbol{r}_j+\boldsymbol{e}'\downarrow}^\dagger c_{\boldsymbol{r}_i\downarrow})\\ &+(\delta_{i+\boldsymbol{e},j+\boldsymbol{e}'}-c_{\boldsymbol{r}_j+\boldsymbol{e}'\uparrow}^\dagger c_{\boldsymbol{r}_i+\boldsymbol{e}\uparrow})(\delta_{i,j}-c_{\boldsymbol{r}_j\downarrow}^\dagger c_{\boldsymbol{r}_i\downarrow})\Big)\end{aligned}$$

3.Attractive Hubbard model

For 1s-wave, $f_\alpha(e=0)=1$

$$\Delta_\alpha^\dagger(\boldsymbol{r}_i)\Delta_\alpha(\boldsymbol{r}_j) = \frac{1}{2}[c^\dagger_{\boldsymbol{r}_i\uparrow}c_{\boldsymbol{r}_j\uparrow}c^\dagger_{\boldsymbol{r}_i\downarrow}c_{\boldsymbol{r}_j\downarrow}]$$

$$\Delta_\alpha(\boldsymbol{r}_i)\Delta_\alpha^\dagger(\boldsymbol{r}_j) = \frac{1}{2}[(\delta_{i,j} - c^\dagger_{\boldsymbol{r}_j\uparrow}c_{\boldsymbol{r}_i\uparrow})(\delta_{i,j} - c^\dagger_{\boldsymbol{r}_j\downarrow}c_{\boldsymbol{r}_i\downarrow})]$$

Only these two terms are necessary for calculating SC correlations.

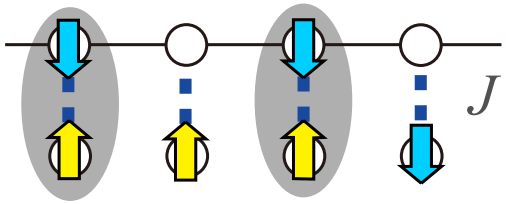
For 2d-wave (dx2-y2), $f_\alpha(e=[1,0])=1$, $f_\alpha(e=[-1,0])=1$, $f_\alpha(e=[0,1])=-1$, $f_\alpha(e=[0,1])=-1$

$$\begin{aligned}\Delta_\alpha^\dagger(\boldsymbol{r}_i)\Delta_\alpha(\boldsymbol{r}_j) = \frac{1}{2}\sum_{\boldsymbol{e},\boldsymbol{e}'} f_\alpha(\boldsymbol{e})f_\alpha(\boldsymbol{e}')\Big(& \\ + c^\dagger_{\boldsymbol{r}_i\uparrow}c_{\boldsymbol{r}_j\uparrow}c^\dagger_{\boldsymbol{r}_i+\boldsymbol{e}\downarrow}c_{\boldsymbol{r}_j+\boldsymbol{e}'\downarrow} & \\ + c^\dagger_{\boldsymbol{r}_i\uparrow}c_{\boldsymbol{r}_j+\boldsymbol{e}'\uparrow}c^\dagger_{\boldsymbol{r}_i+\boldsymbol{e}\downarrow}c_{\boldsymbol{r}_j\downarrow} & \\ + c^\dagger_{\boldsymbol{r}_i+\boldsymbol{e}\uparrow}c_{\boldsymbol{r}_j+\boldsymbol{e}'\uparrow}c^\dagger_{\boldsymbol{r}_i\downarrow}c_{\boldsymbol{r}_j+\boldsymbol{e}'\downarrow} & \\ + c^\dagger_{\boldsymbol{r}_i+\boldsymbol{e}\uparrow}c_{\boldsymbol{r}_j+\boldsymbol{e}'\uparrow}c^\dagger_{\boldsymbol{r}_i\downarrow}c_{\boldsymbol{r}_j\downarrow}\Big) &\end{aligned}$$

$$\begin{aligned}\Delta_\alpha(\boldsymbol{r}_i)\Delta_\alpha^\dagger(\boldsymbol{r}_j) = \frac{1}{2}\sum_{\boldsymbol{e},\boldsymbol{e}'} f_\alpha(\boldsymbol{e})f_\alpha(\boldsymbol{e}')\Big(& \\ + (\delta_{i,j} - c^\dagger_{\boldsymbol{r}_j\uparrow}c_{\boldsymbol{r}_i\uparrow})(\delta_{i+\boldsymbol{e},j+\boldsymbol{e}'} - c^\dagger_{\boldsymbol{r}_j+\boldsymbol{e}'\downarrow}c_{\boldsymbol{r}_i+\boldsymbol{e}\downarrow}) & \\ + (\delta_{i,j+\boldsymbol{e}} - c^\dagger_{\boldsymbol{r}_j+\boldsymbol{e}'\uparrow}c_{\boldsymbol{r}_i\uparrow})(\delta_{i+\boldsymbol{e},j} - c^\dagger_{\boldsymbol{r}_j\downarrow}c_{\boldsymbol{r}_i+\boldsymbol{e}\downarrow}) & \\ + (\delta_{i+\boldsymbol{e},j} - c^\dagger_{\boldsymbol{r}_j\uparrow}c_{\boldsymbol{r}_i+\boldsymbol{e}\uparrow})(\delta_{i,j+\boldsymbol{e}'} - c^\dagger_{\boldsymbol{r}_j+\boldsymbol{e}'\downarrow}c_{\boldsymbol{r}_i\downarrow}) & \\ + (\delta_{i+\boldsymbol{e},j+\boldsymbol{e}'} - c^\dagger_{\boldsymbol{r}_j+\boldsymbol{e}'\uparrow}c_{\boldsymbol{r}_i+\boldsymbol{e}\uparrow})(\delta_{i,j} - c^\dagger_{\boldsymbol{r}_j\downarrow}c_{\boldsymbol{r}_i\downarrow})\Big) &\end{aligned}$$

We should calculate these terms for an-isotropic paring

4. 1D Kondo lattice model



Ex. stan_opt.in

```
L                = 8
Lsub             = 2
lattice          = "chain"
model            = "Kondo"
t                = 1.0
J                = 1.0
ncond            = 8
2Sz              = 0
NVMCSample       = 200
NSROptItrStep    = 600
NMPTrans         = 1
NSPStot          = 0
```

Calculating spin gap from the energy difference between S=0 and S=1 state !

Table 5
Comparisons with exact diagonalization for one-dimensional Kondo-lattice model with $J = 1$, $t = 1$, and $L = 8$. Notations are the same as Table 3. Upper (Lower) panel shows the results for spin singlet (triplet) sector. In the triplet sector ($S = 1$), we take total momentum as $K = \pi$, which gives the lowest energy in $S = 1$, while we take total momentum as $K = 0$ for $S = 1$. The definitions of the spin correlations in the Lanczos method are the same as those of Table 3 for $S = 0$. For $S = 1$, because spin-rotational symmetry is not preserved and S^z correlations are not equivalent to that of S^x and S^y correlations. We do not show the results of the spin correlations in the Lanczos method for $S = 1$.

$(L = 8, S = 0)$	E/N_s	S_{onsite}	$S^{\text{loc}}_{\text{nn}}$	$S(\pi)$	N_p
ED	−1.394104	−0.3151	−0.3386	0.05685	–
mVMC(2)	−1.39350(1)	−0.3144(1)	−0.3363(1)	0.05752(3)	69
mVMC(2)+Lanczos	−1.39401(2)	−0.3152(2)	−0.336(1)	0.05716(4)	69
mVMC(8)	−1.39398(1)	−0.3151(2)	−0.3384(2)	0.05693(4)	261
mVMC(8)+Lanczos	−1.394097(2)	−0.3150(2)	−0.3377(3)	0.0568(1)	261
$(L = 8, S = 1)$	E/N_s	S_{onsite}	$S^{\text{loc}}_{\text{nn}}$	$S(\pi)$	N_p
ED	−1.382061	−0.2748	−0.2240	0.05747	–
mVMC(2)	−1.38126(3)	−0.2738(2)	−0.2246(1)	0.05822(1)	69
mVMC(2)+Lanczos	−1.38187(1)	–	–	–	69
mVMC(8)	−1.38171(3)	−0.2750(4)	−0.2249(7)	0.0577(1)	261
mVMC(8)+Lanczos	−1.382011(2)	–	–	–	261

See, H. Tsunetsugu *et al.*, PRB 46, 3175 (1992),
H. Tsunetsugu *et al.*, RMP 69, 809 (1997)

4. 1D Kondo lattice model

Table 5
Comparisons with exact diagonalization for one-dimensional Kondo-lattice model with $J = 1$, $t = 1$, and $L = 8$. Notations are the same as Table 3. Upper (Lower) panel shows the results for spin singlet (triplet) sector. In the triplet sector ($S = 1$), we take total momentum as $K = \pi$, which gives the lowest energy in $S = 1$, while we take total momentum as $K = 0$ for $S = 1$. The definitions of the spin correlations in the Lanczos method are the same as those of Table 3 for $S = 0$. For $S = 1$, because spin-rotational symmetry is not preserved and S^z correlations are not equivalent to that of S^x and S^y correlations. We do not show the results of the spin correlations in the Lanczos method for $S = 1$.

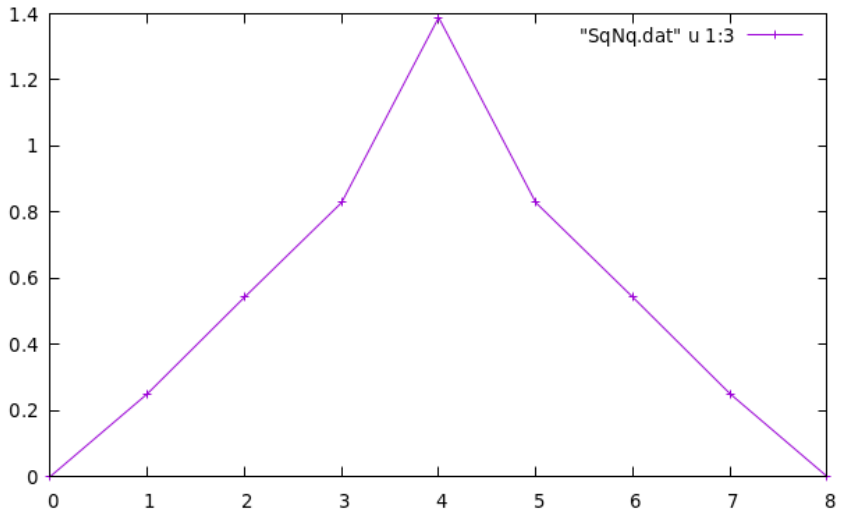
$(L = 8, S = 0)$	E/N_s	S_{onsite}	$S^{\text{loc}}_{\text{nn}}$	$S(\pi)$	N_p
ED	−1.394104	−0.3151	−0.3386	0.05685	–
mVMC(2)	−1.39350(1)	−0.3144(1)	−0.3363(1)	0.05752(3)	69
mVMC(2)+Lanczos	−1.39401(2)	−0.3152(2)	−0.336(1)	0.05716(4)	69
mVMC(8)	−1.39398(1)	−0.3151(2)	−0.3384(2)	0.05693(4)	261
mVMC(8)+Lanczos	−1.394097(2)	−0.3150(2)	−0.3377(3)	0.0568(1)	261
$(L = 8, S = 1)$	E/N_s	S_{onsite}	$S^{\text{loc}}_{\text{nn}}$	$S(\pi)$	N_p
ED	−1.382061	−0.2748	−0.2240	0.05747	–
mVMC(2)	−1.38126(3)	−0.2738(2)	−0.2246(1)	0.05822(1)	69
mVMC(2)+Lanczos	−1.38187(1)	–	–	–	69
mVMC(8)	−1.38171(3)	−0.2750(4)	−0.2249(7)	0.0577(1)	261
mVMC(8)+Lanczos	−1.382011(2)	–	–	–	261

```
$ cat Sij.dat
 # i  j tot_S err_tot_S Sxy err_Sxy Sz err_Sz
0 0   0.75000000   0.00000000   0.50000000   0.00000000   0.25000000   0.00000000
0 1  -0.33608200   0.01193516  -0.22308200   0.00278660  -0.11300000   0.01016735
0 2   0.17766053   0.01250381   0.12416053   0.01278212   0.05350000   0.01190063
0 3  -0.17904993   0.01246032  -0.12404993   0.00538062  -0.05500000   0.01227294
0 4   0.14972480   0.01799936   0.11022480   0.00856900   0.03950000   0.01770240
0 5  -0.17576586   0.01618117  -0.12426586   0.01915799  -0.05150000   0.00752496
0 6   0.15913789   0.00735077   0.11313789   0.00697177   0.04600000   0.00664267
0 7  -0.31820877   0.01514541  -0.21820877   0.01304659  -0.10000000   0.00454148
0 8  -0.30266746   0.01981710  -0.19916746   0.01581060  -0.10350000   0.00456550
0 9   0.09560833   0.01101715   0.06235833   0.00646871   0.03325000   0.00664502
0 10  -0.10533294   0.01317163  -0.07733294   0.01157127  -0.02800000   0.00787202
0 11   0.07217159   0.00547608   0.04592159   0.01023667   0.02625000   0.00733144
0 12  -0.07578796   0.01771080  -0.04953796   0.00726571  -0.02625000   0.01361869
0 13   0.05804883   0.00704554   0.03954883   0.00915289   0.01850000   0.00632949
0 14  -0.06306408   0.02314937  -0.05406408   0.01598994  -0.00900000   0.00799805
0 15   0.09360713   0.00764861   0.07435713   0.00557442   0.01925000   0.00496236
```

4. 1D Kondo lattice model

p "SqNq.dat" u 1:3 w lp

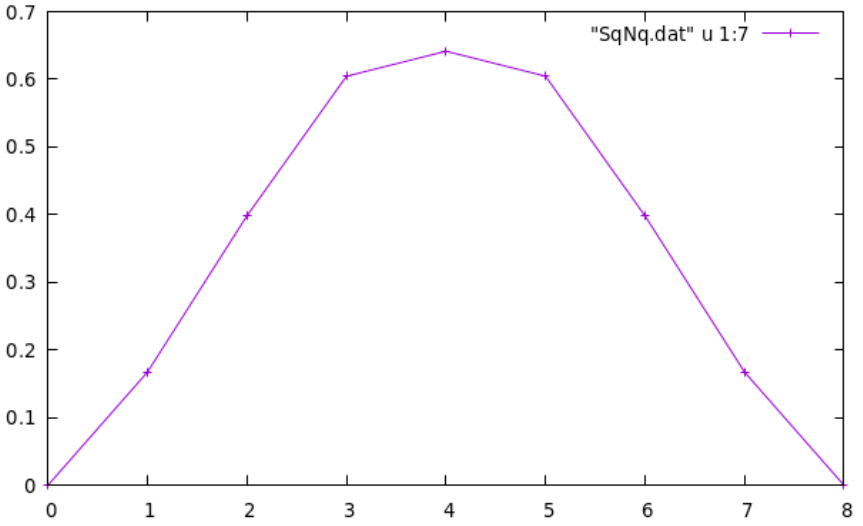
$$S(q) = \frac{1}{N_{\rm s}} \sum_{i,j} \vec{S}_i \cdot \vec{S}_j e^{iqr_i}$$



p "SqNq.dat" u 1:7 w lp

$$N(q) = \frac{1}{N_{\rm s}} \sum_{i,j} \bar{n}_i \cdot \bar{n}_j e^{iqr_i},$$

$$\bar{n}_i = (n_{i\uparrow} + n_{i\downarrow}) - \langle (n_{i\uparrow} + n_{i\downarrow}) \rangle$$



In the Kondo lattice, spin/charge operators are defined as
$S_i = S_i^{f+}S_i^{c}$ and $n_i = n_i^{f}+n_i^{c}$

5. Data repository

Quasi-1D molecular solids TMTTF/TMTSF salts

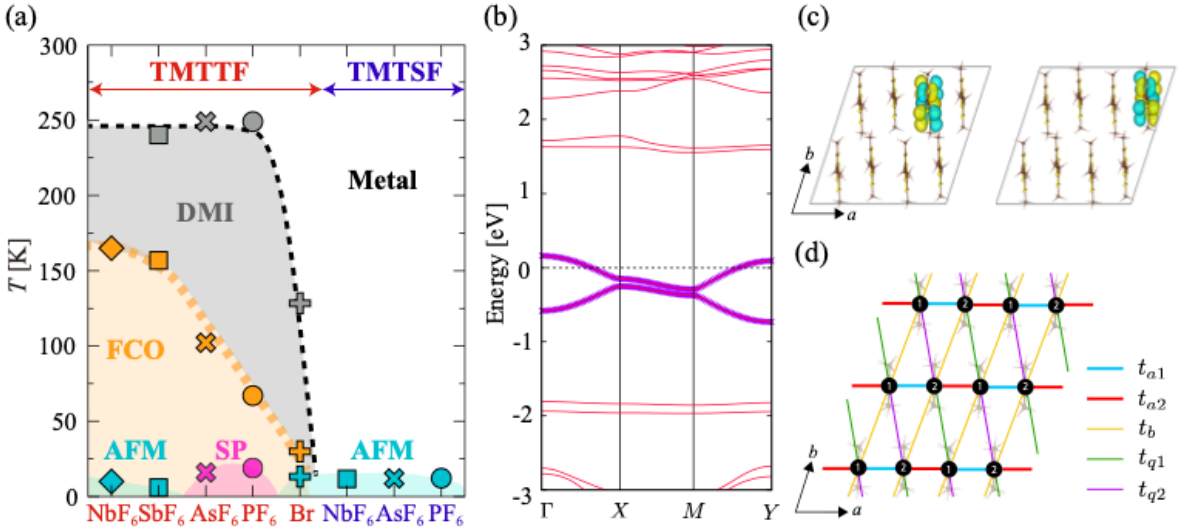
https://isspns-gitlab.issp.u-tokyo.ac.jp/k-yoshimi/tm-salts

- data
  - TMTSF_roomT
    - PF6
      - RESPACK
      - mVMC
      - qe
    - AsF6
      - RESPACK
      - mVMC
      - qe
    - SbF6
      - RESPACK
      - mVMC
      - qe

第一原理計算の結果(QE)

有効模型導出の結果(RESPACK)

有効模型解析結果(mVMC)



https://arxiv.org/abs/2210.13726

5. Data repository

Correlated Dirac electrons in α-ET and α-BETS

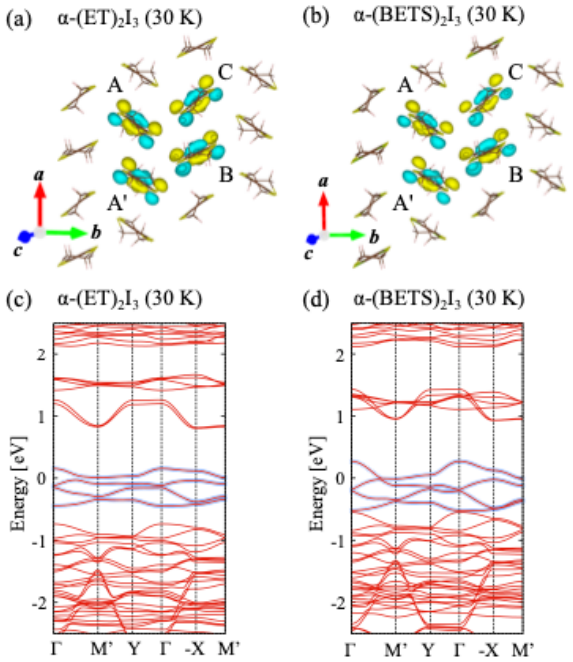
https://isspns-gitlab.issp.u-tokyo.ac.jp/k-yoshimi/alpha-salts

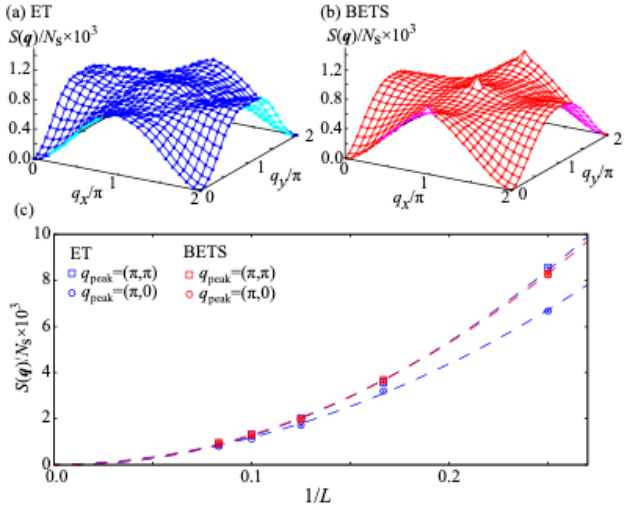
- data
  - alpha-BETS
    - 30K
      - RESPACK
      - mVMC
      - qe
    - 80K
      - RESPACK
      - qe
  - alpha-ET
    - 150K
      - RESPACK
      - mVMC
      - qe
    - 30K
      - RESPACK
      - mVMC
      - qe

第一原理計算の結果(QE)

有効模型導出の結果(RESPACK)

有効模型解析結果(mVMC)





https://arxiv.org/abs/2209.13460