Sep 15, 2025

Selected Topics in Computational Quantum Physics

量子物理计算方法选讲

Shuo Yang（杨硕）

Department of Physics, Tsinghua University

Email: shuoyang@tsinghua.edu.cn

WeChat: condmat-ys

该二维码7天内(9月22日前)有效，重新进入将更新

Goals of this course

- bridge the gap between standard undergraduate quantum mechanics course and frontier research
- introduction of topics, methods, and algorithms that are commonly used in modern condensed matter physics and quantum physics
- explain, implement, and apply computational methods to study quantum many-body systems
- evaluate, analyze, and interpret numerical simulation results
- deeper understanding at the qualitative and quantitate level
- share the colourfulness, beauty, and excitement of quantum many-body physics
- foster free, independent, and critical thinking

Features of this course

- focus on strongly correlated many-body systems defined on a lattice
- lots of original data and numerical observations
- along the historical path of discovery and understanding
- some exposure to algorithms and coding techniques
- paves the way for further theoretical study and research
- teaching and learning through Q & A
- hands-on tutorials and sample codes

Requirements of this course

- basic knowledge of quantum mechanics, solid state physics, and programming
- active participation
- extensive reading and intensive discussion
- academic integrity

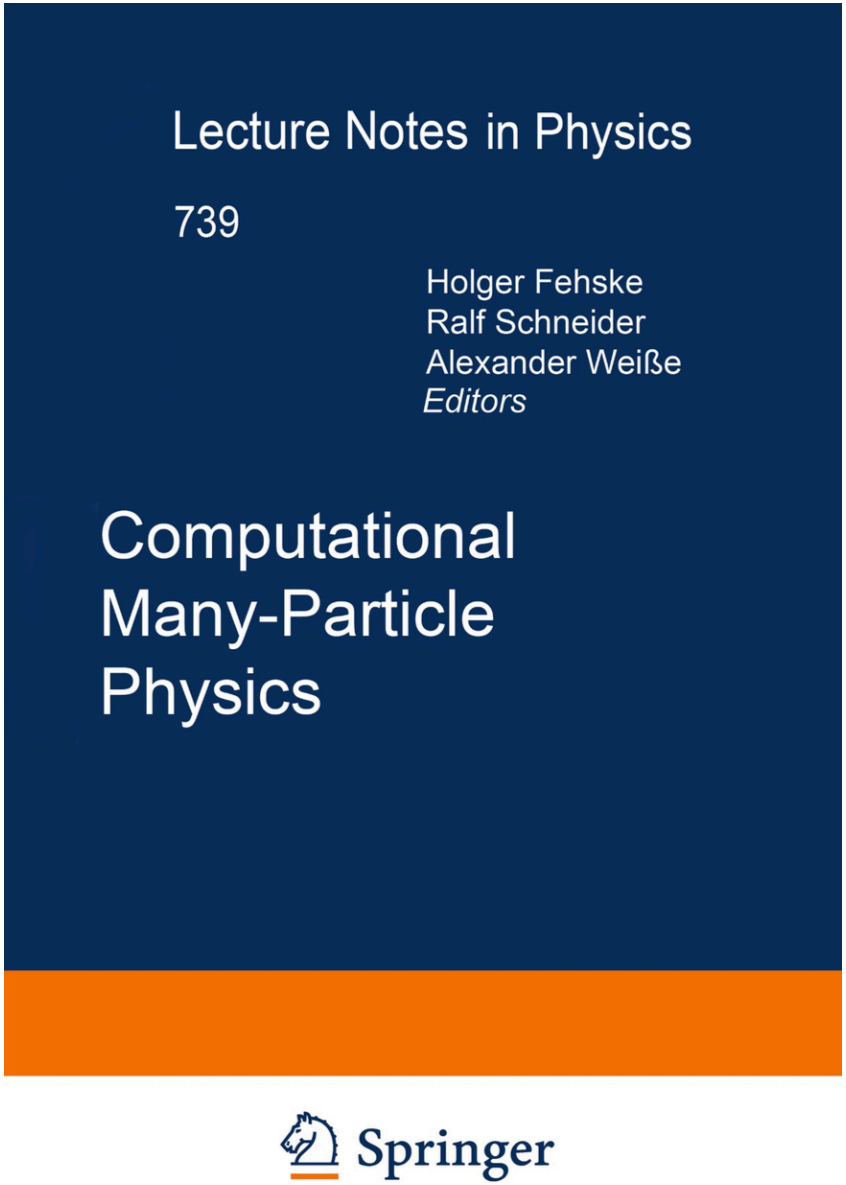
Grading of the course

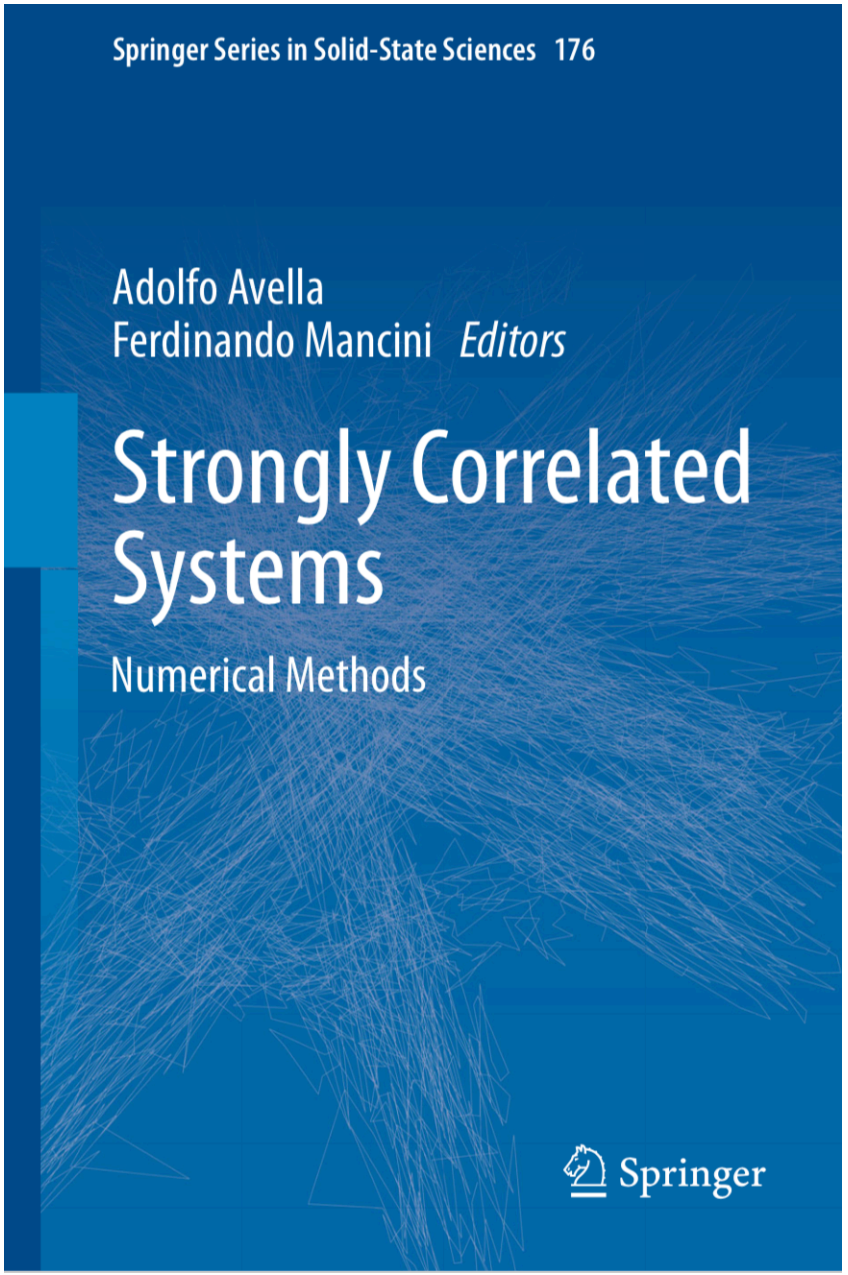
- homework: 80%, finial project: 20%, involvement: 10%
- A-to-F scale system

References of the course

- can be found online http://learn.tsinghua.edu.cn/

References of the course





Strongly correlated quantum many-body system

- what is “strongly correlated”?

  single particle approximation fails due to interactions

- examples of strongly correlated systems?

  high temperature superconductors, fractional quantum Hall systems, quantum magnetism, ultra-cold atom gases …

- what do we study mainly?

  ground state, low excitations, physical observables, time evolution, finite temperature …

Strongly correlated quantum many-body system

- what are the main challenges?

  no exact solution, Hilbert space too large, 2D & 3D, sign problem, violate area law …

- how to solve the problem?

  theoretically: mean field approximation, projected wave function, functional RG …

  numerically: DFT+U, ED, DMRG, QMC, NRG, DMFT, Tensor networks, Machine learning …

- features of computational quantum many-body physics

  a wide variety of topics, rapid progress, close interactions with theory and experiments

Outline of the course

- Chapter 0: Introduction

  brief review of quantum mechanics, microscopic lattice models, quantum phase transition, matrix operations, python programming

- Chapter 1: Exact Diagonalization

  matrix representation of Hamiltonian, Lanczos method, time evolution, momentum space exact diagonaliation, various applications

- Chapter 2: Density Matrix Renormalization Group and Matrix Product States

  traditional DMRG method, many-body entanglement, matrix product state and its symmetry, iTEBD method, variational MPS method

Outline of the course

- Chapter 3: Tensor Networks

  projected entangled-pair states (PEPS), multi-scale entanglement renormalization ansatz (MERA), tensor network and topological order, tensor renormalization group, 2D algorithms and applications

- Chapter 4: Quantum Monte Carlo

  important sampling, classical Monte Carlo, fermion sign problem, various Quantum Monte Carlo methods including Determinate QMC, Path-Integral QMC, Variational QMC, Stochastic Series Expansion QMC, Majorana QMC …

Outline of the course

- Chapter 5: Introduction to Other Methods

  numerical renormalization group (NRG), dynamical mean field theory (DMFT), mechine learning and its new progress in physics

- Chapter 6: Summary and Discussions

  universality and finite size scaling, comparison of different methods, challenges and opportunities for computational quantum many-body physics

Basis of quantum mechanics

- quantum mechanics is nothing but simple linear algebra
  albeit in huge Hilbert spaces, which makes the problem hard

- wave function $|\Psi\rangle$
  scalar product of two wave functions $\langle\Phi|\Psi\rangle$

- spin-1/2 system, basis vectors are the "up" and "down" spin states

$$|\uparrow\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad |\downarrow\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

  quantum spin can exist in any complex superposition

$$|\Psi\rangle = \alpha|\uparrow\rangle + \beta|\downarrow\rangle$$

  normalization condition requires that $|\alpha|^2 + |\beta|^2 = 1$

- for example, a spin pointing in the positive x-direction is described by

$$|\rightarrow\rangle = \frac{1}{\sqrt{2}}(|\uparrow\rangle + |\downarrow\rangle)$$

Mixed states and density matrices

- unless specifically prepared in a pure state in an experiment, quantum systems in nature rarely exist as pure states but instead as probabilistic superpositions

- the most general state of a quantum system is described as a density matrix ρ with unit trace $\mathrm{Tr}\,\rho = 1$

- the density matrix of a pure state is $\rho_{\text{pure}} = |\Psi\rangle\langle\Psi|$

  the density matrix of a mixed state is $\rho = \sum_i p_i \left|\psi_i\right\rangle \left\langle\psi_i\right|$

- density matrix of a spin pointing in the positive x-direction

$$\rho_{\rightarrow} = |\rightarrow\rangle\langle\rightarrow| = \begin{pmatrix} 1/2 & 1/2 \\ 1/2 & 1/2 \end{pmatrix}$$

  probabilistic mixed state with 50% probability $|0\rangle$ and 50% probability $|1\rangle$

$$\rho = \frac{1}{2}|0\rangle\langle 0| + \frac{1}{2}|1\rangle\langle 1| = \begin{pmatrix} 1/2 & 0 \\ 0 & 1/2 \end{pmatrix}$$

The measurement process

- the outcome of a measurement in a quantum system is usually intrusive and not deterministic

- after measuring an observable A, the new wave function of the system will be an eigenvector of A and the outcome of the measurement the corresponding eigenvalue

- we start with a spin pointing up with wave function $|\Psi\rangle = |\uparrow\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$
  or alternatively density matrix $\rho_\uparrow = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$

- we measure the x-component of the spin S^x

  result: $+1/2$ wave function: $|\rightarrow\rangle = (|\uparrow\rangle + |\downarrow\rangle)/\sqrt{2}$

  result: $-1/2$ wave function: $|\leftarrow\rangle = (|\uparrow\rangle - |\downarrow\rangle)/\sqrt{2}$

- the final state differs from the initial density matrix

$$\rho = p_\rightarrow |\rightarrow\rangle\langle\rightarrow| + p_\leftarrow |\leftarrow\rangle\langle\leftarrow| = \begin{pmatrix} 1/2 & 0 \\ 0 & 1/2 \end{pmatrix}$$

The measurement process

- if we are not interested in the result of a particular outcome but just in the average outcome, the expectation value can be calculated from a wave function $|\Psi\rangle$ as $\langle A\rangle = \langle\Psi|A|\Psi\rangle$

  or from a density matrix ρ as $\langle A\rangle = \mathrm{Tr}(\rho A)$

- if we want to describe a physical system not in the ground state but in thermal equilibrium at a given inverse temperature $\beta = 1/k_B T$

- in a classical system each microstate i of energy E_i occupied with a probability given by the Boltzman distribution

  $$p_i = \frac{1}{Z}\exp\left(-\beta E_i\right) \qquad \text{with the partition function} \qquad Z = \sum_i \exp\left(-\beta E_i\right)$$

- in a quantum system, the density matrix is

  $$\rho_\beta = \frac{1}{Z}\exp(-\beta H) \qquad \text{with the partition function} \qquad Z = \mathrm{Tr}\exp(-\beta H)$$

  the thermal average of an observable A is

  $$\langle A\rangle = \mathrm{Tr}\left(A\rho_\beta\right) = \frac{\mathrm{Tr}\, A\exp(-\beta H)}{\mathrm{Tr}\exp(-\beta H)}$$

The Schroedinger equation

- the wave function $|\Psi\rangle$ of a quantum system evolves according to

$$i\hbar\frac{\partial}{\partial t}|\Psi(t)\rangle = H|\Psi(t)\rangle$$

this is the time-dependent Schoredinger equation

- for a stationary time-independent problem, using the ansatz

$$|\Psi(t)\rangle = \exp(-iEt/\hbar)|\Psi\rangle$$

the Schroedinger equation simplifies to a linear eigenvalue problem

$$\boxed{H|\Psi\rangle = E|\Psi\rangle}$$

the rest semester will be spent solving just this simple eigenvalue problem

- the Schrodinger equation for the density matrix

$$i\hbar\frac{\partial}{\partial t}\rho(t) = [H, \rho(t)]$$

Second quantization

- occupation number representation

  an arbitrary many-particle basis state with n_1 particles in single-particle state 1, n_2 particles in state 2, n_3 particles in state 3 etc.

$$\left| n_1, n_2, n_3, n_4, n_5, \ldots \right\rangle$$

- bosonic creation operator

$$a_\alpha^\dagger \left| n_1, n_2, \ldots, n_{\alpha-1}, n_\alpha, n_{\alpha+1}, \ldots \right\rangle = \sqrt{n_\alpha + 1} \left| n_1, n_2, \ldots, n_{\alpha-1}, n_\alpha + 1, n_{\alpha+1}, \ldots \right\rangle$$

  bosonic annihilation operator

$$a_\alpha \left| n_1, n_2, \ldots, n_{\alpha-1}, n_\alpha, n_{\alpha+1}, \ldots \right\rangle = \sqrt{n_\alpha} \left| n_1, n_2, \ldots, n_{\alpha-1}, n_\alpha - 1, n_{\alpha+1}, \ldots \right\rangle$$

- fermionic creation operator

$$c_\alpha^\dagger \left| n_1, n_2, \ldots, n_{\alpha-1}, 0_\alpha, n_{\alpha+1}, \ldots \right\rangle = (-1)^{\nu_\alpha} \left| n_1, n_2, \ldots, n_{\alpha-1}, 1_\alpha, n_{\alpha+1}, \ldots \right\rangle, \quad \nu_\alpha = \sum_{\beta<\alpha} c_\beta^\dagger c_\beta = \sum_{\beta<\alpha} n_\beta$$

$$c_\alpha^\dagger \left| n_1, n_2, \ldots, n_{\alpha-1}, 1_\alpha, n_{\alpha+1}, \ldots \right\rangle = 0$$

  fermionic annihilation operator

$$c_\alpha \left| n_1, n_2, \ldots, n_{\alpha-1}, 1_\alpha, n_{\alpha+1}, \ldots \right\rangle = (-1)^{\nu_\alpha} \left| n_1, n_2, \ldots, n_{\alpha-1}, 0_\alpha, n_{\alpha+1}, \ldots \right\rangle, \quad \nu_\alpha = \sum_{\beta<\alpha} c_\beta^\dagger c_\beta = \sum_{\beta<\alpha} n_\beta$$

$$c_\alpha \left| n_1, n_2, \ldots, n_{\alpha-1}, 0_\alpha, n_{\alpha+1}, \ldots \right\rangle = 0$$

Second quantization

- commutation relations for bosons

$$\left[a_\alpha^\dagger, a_\beta^\dagger\right] = \left[a_\alpha, a_\beta\right] = 0, \quad \left[a_\alpha, a_\beta^\dagger\right] = \delta_{\alpha,\beta}$$

where $[A, B] = AB - BA$

example $|2,1,0,0,\ldots\rangle \sim a_1^\dagger a_1^\dagger a_2^\dagger |0\rangle = a_1^\dagger a_2^\dagger a_1^\dagger |0\rangle = a_2^\dagger a_1^\dagger a_1^\dagger |0\rangle$

- anti-commutation relations for fermions

$$\left\{c_\alpha^\dagger, c_\beta^\dagger\right\} = \left\{c_\alpha, c_\beta\right\} = 0, \quad \left\{c_\alpha, c_\beta^\dagger\right\} = \delta_{\alpha,\beta}$$

where $\{A, B\} = AB + BA$

example $\left|n_1, n_2, \ldots, n_\alpha, \ldots\right\rangle = \left(c_1^\dagger\right)^{n_1}\left(c_2^\dagger\right)^{n_2}\ldots\left(c_\alpha^\dagger\right)^{n_\alpha}\ldots|0\rangle$

$$|1,1,1\rangle = c_1^\dagger c_2^\dagger c_3^\dagger |0\rangle = -c_2^\dagger c_1^\dagger c_3^\dagger |0\rangle = c_2^\dagger c_3^\dagger c_1^\dagger |0\rangle = -c_3^\dagger c_2^\dagger c_1^\dagger |0\rangle = c_3^\dagger c_1^\dagger c_2^\dagger |0\rangle = -c_1^\dagger c_3^\dagger c_2^\dagger |0\rangle$$

$$c_2^\dagger c_1^\dagger |0\rangle = c_2^\dagger \left|1_1,0_2,0_3,\ldots\right\rangle = -\left|1_1,1_2,0_3,\ldots\right\rangle$$

Tensor product of quantum system

- combining two systems A and B

  the basis of A is $|A_0\rangle$ and $|A_1\rangle$, the basis of B is $|B_0\rangle$ and $|B_1\rangle$

  the set of basis states for the combined system is

$$|C_{00}\rangle = |A_0\rangle\,|B_0\rangle,\ |C_{01}\rangle = |A_0\rangle\,|B_1\rangle,\ |C_{10}\rangle = |A_1\rangle\,|B_0\rangle,\ |C_{11}\rangle = |A_1\rangle\,|B_1\rangle\,.$$

- if A is in the state $\quad |\psi_A\rangle = a_0\,|A_0\rangle + a_1\,|A_1\rangle$

  and B is in the state $\quad |\psi_B\rangle = b_0\,|B_0\rangle + b_1\,|B_1\rangle$

  the combined state is $\quad |\psi_C\rangle = \sum_{ij} c_{ij}\,|C_{ij}\rangle = |\psi_A\rangle \otimes |\psi_B\rangle$

$$|\psi_C\rangle = \begin{pmatrix} a_0 \begin{pmatrix} b_0 \\ b_1 \end{pmatrix} \\ a_1 \begin{pmatrix} b_0 \\ b_1 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} a_0 b_0 \\ a_0 b_1 \\ a_1 b_0 \\ a_1 b_1 \end{pmatrix} = \begin{pmatrix} c_{00} \\ c_{01} \\ c_{10} \\ c_{11} \end{pmatrix}$$

- it is called outer product for vectors or Kronecker product for matrices

  https://docs.scipy.org/doc/numpy/reference/generated/numpy.kron.html

Tensor product of quantum system

- tensor product for quantum operators

$$\begin{bmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{bmatrix} \otimes \begin{bmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \end{bmatrix} = \begin{bmatrix} a_{1,1}\begin{bmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \end{bmatrix} & a_{1,2}\begin{bmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \end{bmatrix} \\ a_{2,1}\begin{bmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \end{bmatrix} & a_{2,2}\begin{bmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \end{bmatrix} \end{bmatrix} = \begin{bmatrix} a_{1,1}b_{1,1} & a_{1,1}b_{1,2} & a_{1,2}b_{1,1} & a_{1,2}b_{1,2} \\ a_{1,1}b_{2,1} & a_{1,1}b_{2,2} & a_{1,2}b_{2,1} & a_{1,2}b_{2,2} \\ a_{2,1}b_{1,1} & a_{2,1}b_{1,2} & a_{2,2}b_{1,1} & a_{2,2}b_{1,2} \\ a_{2,1}b_{2,1} & a_{2,1}b_{2,2} & a_{2,2}b_{2,1} & a_{2,2}b_{2,2} \end{bmatrix}$$

- the tensor product of (finite-dimensional) vector spaces has dimension equal to the product of the dimensions of the two factors

$$\dim(V \otimes W) = \dim V \times \dim W$$

- a single spin-1/2 has a Hilbert space of dimension 2

  N spin-1/2 have a Hilbert space of dimension 2^N

  for example, $2^{30} \approx 10^9$

  this shows the complexity of the quantum many body problem!

Microscopic lattice models

- Hubbard model

$$H = -t \sum_{\langle i,j \rangle, \sigma} c_{i\sigma}^{\dagger} c_{j\sigma} + U \sum_{i} n_{i\uparrow} n_{i\downarrow}$$

- Heisenberg model

$$H = J \sum_{\langle i,j \rangle} \mathbf{S}_i \cdot \mathbf{S}_j = J \sum_{\langle i,j \rangle} \left(S_i^x S_j^x + S_i^y S_j^y + S_i^z S_j^z \right)$$

$$S^x = \frac{1}{2}\sigma^x = \frac{1}{2}\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, S^y = \frac{1}{2}\sigma^y = \frac{1}{2}\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, S^z = \frac{1}{2}\sigma^z = \frac{1}{2}\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}.$$

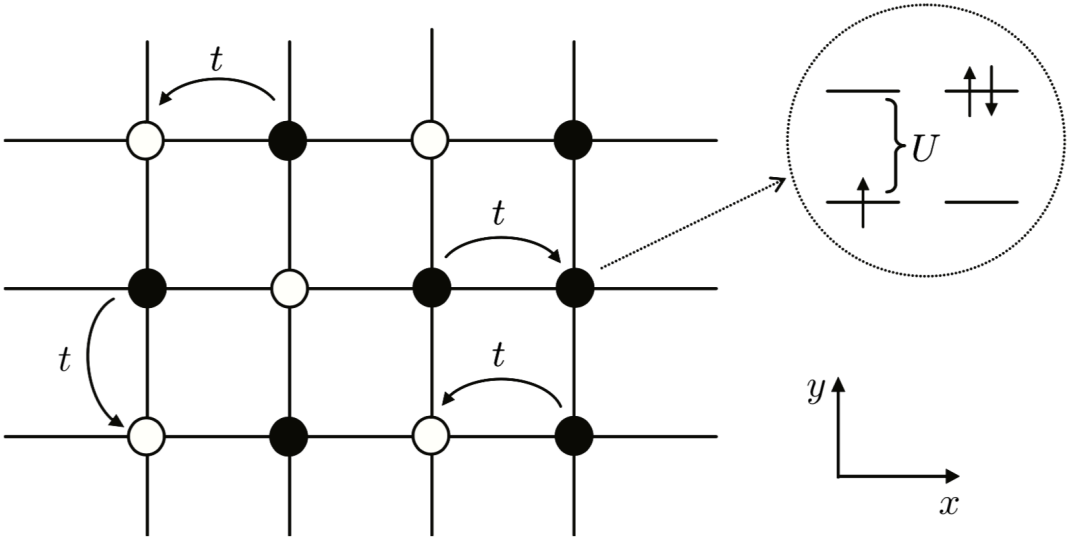
- transverse field Ising model

$$H = -J \sum_{i=1}^{N} \left(g\sigma_i^x + \sigma_i^z \sigma_{i+1}^z \right)$$

The Hubbard model

- important model in solid state theory, formulated by John Hubbard

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



- the first term describes the hopping of electrons between two neighboring sites, called hopping term
- the 2nd term describes the onsite Coulomb repulsion between two electrons occupying the same site, the on-site Coulomb repulsion.

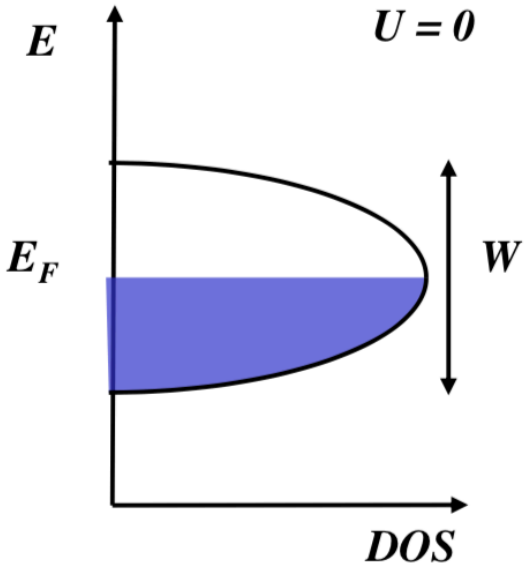
- this is the simplest model of interacting particles in a lattice

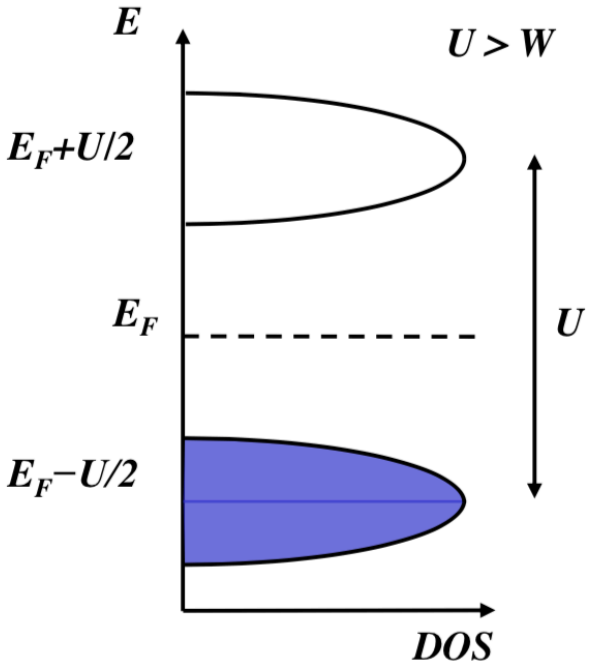
  simple model describing many phenomena

  no general solution exist but limiting cases well understood (Numerics!)

Band structure of the Hubbard Model

$$H = -t \sum_{\langle i,j \rangle,\sigma} c_{i\sigma}^{\dagger} c_{j\sigma} + U \sum_{i} n_{i\uparrow} n_{i\downarrow}$$





- Hubbard model reduced to tight binding model

  bandwidth $W \sim 2zt >> U$

- half-filled band, metallic ground state

- onsite Coulomb repulsion splits the band into two halves

  the lower band is full, the upper band is empty

- the ground state is a Mott insulator

The Heisenberg model

- when $U >> t$, at half filling, Hubbard model is reduced to Heisenberg model

$$H = J\sum_{\langle i,j\rangle} \mathbf{S}_i \cdot \mathbf{S}_j = J\sum_{\langle i,j\rangle} \left(S_i^x S_j^x + S_i^y S_j^y + S_i^z S_j^z \right)$$

$$= J\sum_{\langle i,j\rangle} \left[\left(S_i^+ S_j^- + S_i^- S_j^+ \right)/2 + S_i^z S_j^z \right]$$

- $J < 0$, exact ground state is trivial, fully polarized ferromagnetic ordering

$$|\uparrow\uparrow\uparrow\uparrow\uparrow\uparrow\uparrow\uparrow\rangle$$

- $J > 0$, naive (classical) guess is not an eigenstate

$$\left|\psi_{\text{Neel}}\right\rangle = |\uparrow\downarrow\uparrow\downarrow\uparrow\downarrow\uparrow\downarrow\uparrow\downarrow\rangle$$

off-diagonal terms $S_i^+ S_j^- + S_i^- S_j^+$ exchange spins

$$|\uparrow\downarrow\uparrow\downarrow\uparrow\downarrow\uparrow\downarrow\rangle \rightarrow |\uparrow\downarrow\uparrow\downarrow\downarrow\uparrow\uparrow\downarrow\rangle$$

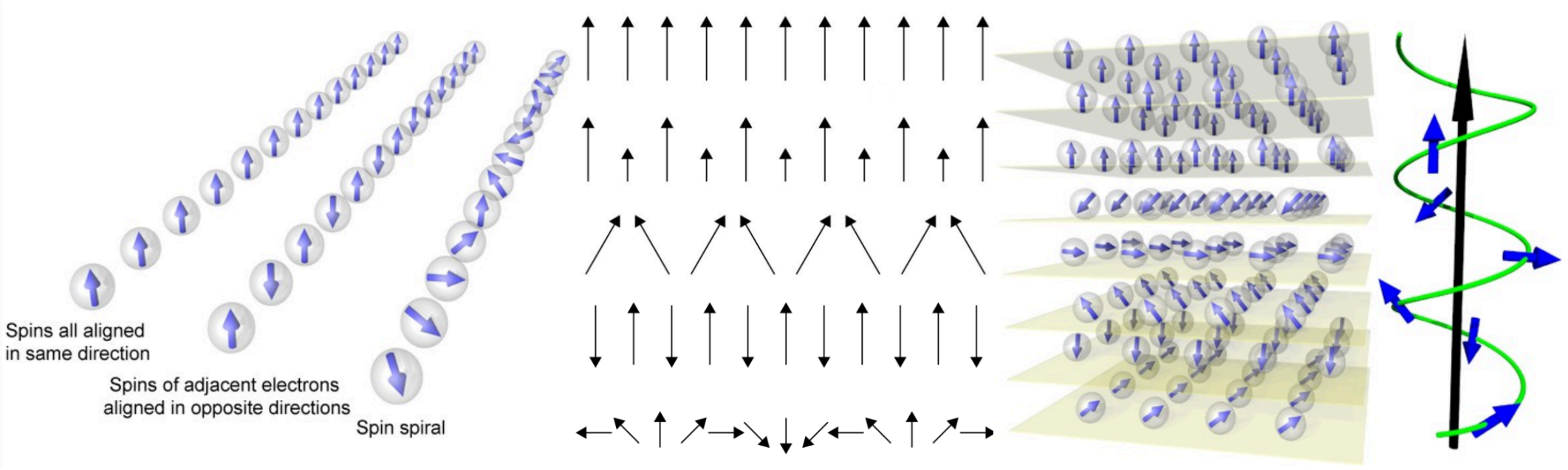
- finding the ground state is surprisingly difficult
  very rich and not get fully understood physics can arise

More general Heisenberg model

$$H = \sum_{i,j} J_{i,j} \mathbf{S}_i \cdot \mathbf{S}_j$$

- non-negligible *J'* for more distant neighbors or in geometries leading to magnetic frustrations → more complicated magnetic structures

- e.g., spin spirals, non-collinear structures …



Transverse field Ising model

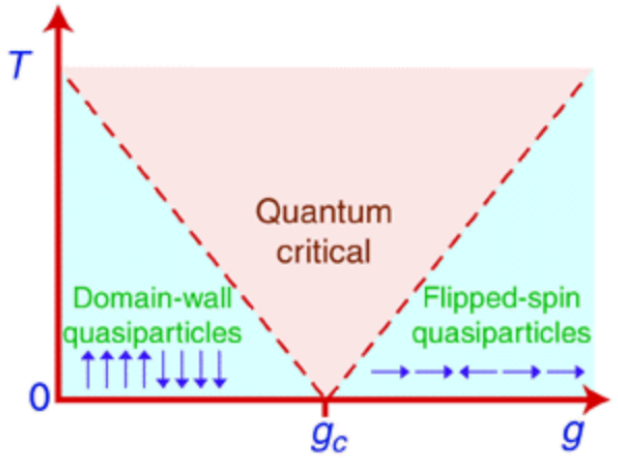
$$H = -J\sum_{i=1}^{N}\left(g\sigma_i^x + \sigma_i^z\sigma_{i+1}^z\right) \qquad J, g \geq 0$$

$$\sigma^x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \sigma^z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

- the simplest exactly solvable model with quantum phase transition
- quantum phase transition

  phase transitions at zero temperature, which occur when some parameter in the Hamiltonian is varied
- ground states in two limits

$$|G\rangle = \begin{cases} |\uparrow\uparrow\cdots\uparrow\rangle \text{ or } |\downarrow\downarrow\cdots\downarrow\rangle & g = 0 \\ |\rightarrow\rightarrow\cdots\rightarrow\rangle & g \to \infty \end{cases}$$



- $g_c = 1$ is the phase transition point
- see the book "Quantum phase transition" by Sachdev, chapter 4.2 for details

Introduction to Python

Python is a high-level, interpreted and general-purpose dynamic programming language that focuses on code readability.

- very rich scientific computing libraries, such as Numpy and Scipy
- well thought out language, allowing to write very readable and well structured code: we “code what we think”
- many libraries beyond scientific computing (web server, serial port access, etc)
- free and open-source software, widely spread, with a vibrant community
- a variety of powerful environments to work in, such as IPython, Spyder, Jupyter notebooks, Pycharm

Useful matrix decompositions

- Eigen-decomposition

$$A = V\Lambda V^{-1}$$

numpy.linalg.eig, numpy.linalg.eigh

scipy.linalg.eig, scipy.sparse.linalg.eigs, scipy.sparse.linalg.eigsh

- Singular Value Decomposition (SVD)

$$A = USV$$

numpy.linalg.svd, scipy.sparse.linalg.svds

- QR and LQ decomposition

$$A = QR \qquad A = LQ$$

numpy.linalg.qr, scipy.linalg.qr

Eigen-decomposition

- eigenvalue problem

  a (non-zero) vector v of dimension N is an eigenvector of a square $N \times N$ matrix A if it satisfies the linear equation

  $$Av = \lambda v$$

  where λ is a scalar, termed the eigenvalue corresponding to v

- example $A = \begin{pmatrix} 5 & 1 \\ 3 & 3 \end{pmatrix}, \lambda_1 = 6, v_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \lambda_2 = 2, v_2 = \begin{pmatrix} 1 \\ -3 \end{pmatrix}$

- eigen-decomposition of a matrix

  let A be a square $N \times N$ matrix with N linearly independent eigenvectors v_i (where $i = 1, ..., N$). Then A can be factorized as

  $$A = V \Lambda V^{-1}$$

  where V is the square $N \times N$ matrix whose i-th column is the eigenvector v_i of A, and Λ is the diagonal matrix whose diagonal elements are the corresponding eigenvalues, $\Lambda_{ii} = \lambda_i$

- example $V = \begin{pmatrix} 1 & 1 \\ 1 & -3 \end{pmatrix}, \Lambda = \begin{pmatrix} 6 & 0 \\ 0 & 2 \end{pmatrix}$

numpy.linalg.eig

`linalg.eig(a)` [source]

Compute the eigenvalues and right eigenvectors of a square array.

Parameters: **a** : *(..., M, M) array*

Matrices for which the eigenvalues and right eigenvectors will be computed

Returns: **w** : *(..., M) array*

The eigenvalues, each repeated according to its multiplicity. The eigenvalues are not necessarily ordered. The resulting array will be of complex type, unless the imaginary part is zero in which case it will be cast to a real type. When *a* is real the resulting eigenvalues will be real (0 imaginary part) or occur in conjugate pairs

v : *(..., M, M) array*

The normalized (unit "length") eigenvectors, such that the column `v[:,i]` is the eigenvector corresponding to the eigenvalue `w[i]`.

```
In [32]: import numpy as np
         import numpy.linalg as LA
```

```
In [33]: A=np.asarray([[5,1],[3,3]])
         Lambda,U = LA.eig(A)
```

```
In [34]: print(A)
         print(U)
         print(Lambda)
```

```
[[5 1]
 [3 3]]
[[ 0.70710678 -0.31622777]
 [ 0.70710678  0.9486833 ]]
[6. 2.]
```

```
In [40]: print(np.dot(np.dot(V,np.diag(Lambda)),LA.inv(V)))
```

```
[[5. 1.]
 [3. 3.]]
```

$$A = V\Lambda V^{-1}$$

$$A = \begin{pmatrix} 5 & 1 \\ 3 & 3 \end{pmatrix}$$

$$V = \begin{pmatrix} 1 & 1 \\ 1 & -3 \end{pmatrix}$$

$$\Lambda = \begin{pmatrix} 6 & 0 \\ 0 & 2 \end{pmatrix}$$

Eigen-decomposition

- generalized eigenvalue problem

  finding a vector v that obeys

  $$Av = \lambda Bv$$

  where A and B are matrices

- complex Hermitian or real symmetric matrix

  $$A = U\Lambda U^{\dagger} \longleftrightarrow \Lambda = U^{\dagger}AU$$

  where U is a unitary matrix

  $$U^{-1} = U^{\dagger} \qquad\qquad UU^{\dagger} = U^{\dagger}U = I$$

  example

$$A = \begin{pmatrix} 6 & 2 \\ 2 & 3 \end{pmatrix}, U = \begin{pmatrix} 0.8944 & -0.4472 \\ 0.4472 & 0.8944 \end{pmatrix}, \Lambda = \begin{pmatrix} 7 & 0 \\ 0 & 2 \end{pmatrix}, U^{\dagger} = \begin{pmatrix} 0.8944 & 0.4472 \\ -0.4472 & 0.8944 \end{pmatrix}$$

numpy.linalg.eigh

```
linalg.eigh(a, UPLO='L')
```

Returns:

w : *(..., M) ndarray*

The eigenvalues in ascending order, each repeated according to its multiplicity.

v : *{(..., M, M) ndarray, (..., M, M) matrix}*

The column `v[:, i]` is the normalized eigenvector corresponding to the eigenvalue `w[i]`. Will return a matrix object if *a* is a matrix object.

In [70]:
```
import numpy as np
import numpy.linalg as LA
```

In [71]:
```
A=np.asarray([[6,2],[2,3]])
Lambda,U = LA.eigh(A)
```

In [72]:
```
print(A)
print(U)
print(Lambda)
```

```
[[6 2]
 [2 3]]
[[ 0.4472136  -0.89442719]
 [-0.89442719 -0.4472136 ]]
[2. 7.]
```

In [73]:
```
print(np.dot(U,np.transpose(np.conj(U))))
print(np.dot(np.transpose(np.conj(U)),U))
print(LA.inv(U)-np.transpose(np.conj(U)))
```

```
[[1. 0.]
 [0. 1.]]
[[1.00000000e+00 1.25949234e-17]
 [1.25949234e-17 1.00000000e+00]]
[[ 0.00000000e+00 -1.11022302e-16]
 [ 0.00000000e+00  0.00000000e+00]]
```

$$UU^{\dagger} = U^{\dagger}U = I$$

$$U^{-1} = U^{\dagger}$$

In [74]:
```
print(np.dot(np.dot(U,np.diag(Lambda)),np.transpose(np.conj(U))))
```

```
[[6. 2.]
 [2. 3.]]
```

$$A = U\Lambda U^{\dagger}$$

Singular Value Decomposition (SVD)

- for an arbitrary matrix A of dimension $(m \times n)$ there existence of a decomposition

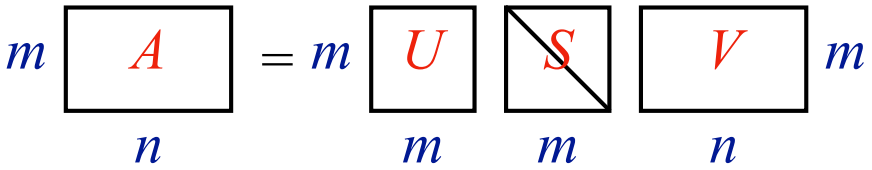
$$A = U\,S\,V$$

where S is diagonal, and U and V are unitary matrices

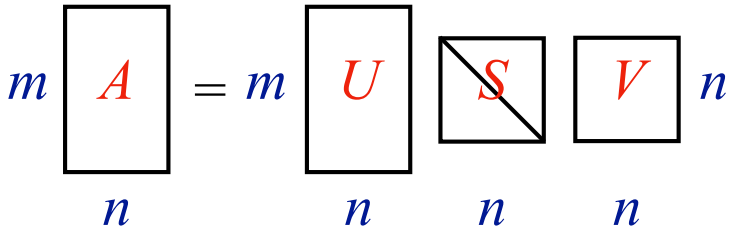
$$S = \text{diag}\left(s_1, s_2, \cdots, s_r\right), s_1 \geq s_2 \geq \cdots \geq s_r \geq 0, r = \min(m, n)$$

$$UU^\dagger = U^\dagger U = \mathbb{I}, VV^\dagger = V^\dagger V = \mathbb{I} \quad \text{when } m = n$$

- when $m < n$



- when $m > n$



- when $m < n$ $\quad UU^\dagger = U^\dagger U = VV^\dagger = \mathbb{I}$, $V^\dagger V \neq \mathbb{I}$

```
In [75]: import numpy as np
         import numpy.linalg as LA

In [76]: A = np.random.rand(2,3)
         U,S,V = LA.svd(A,full_matrices=0)

In [77]: print(A)
         print(U)
         print(S)
         print(V)
```

```
[[0.81295543 0.33147708 0.59651573]
 [0.44628411 0.66699245 0.8255719 ]]
[[ 0.67377523 -0.73893636]
 [ 0.73893636  0.67377523]]
[1.51890393 0.38098931]
[[ 0.57773555  0.47152818  0.66624531]
 [-0.78749492  0.5366621   0.3030603 ]]
```

```
In [78]: print(np.dot(U,np.transpose(np.conj(U))))
         print(np.dot(np.transpose(np.conj(U)),U))
         print(np.dot(V,np.transpose(np.conj(V))))
         print(np.dot(np.transpose(np.conj(V)),V))
```

```
[[1. 0.]
 [0. 1.]]
[[1.0000000e+00 1.3212057e-18]
 [1.3212057e-18 1.0000000e+00]]
[[1.00000000e+00 3.33066907e-16]
 [3.33066907e-16 1.00000000e+00]]
[[ 0.95392661 -0.15020008  0.14625515]
 [-0.15020008  0.51034504  0.47679442]
 [ 0.14625515  0.47679442  0.53572835]]
```

- when $m > n$ $\quad U^\dagger U = VV^\dagger = V^\dagger V = \mathbb{I}$, $UU^\dagger \neq \mathbb{I}$

```
In [75]: import numpy as np
         import numpy.linalg as LA

In [79]: A = np.random.rand(3,2)
         U,S,V = LA.svd(A,full_matrices=0)

In [80]: print(A)
         print(U)
         print(S)
         print(V)
```

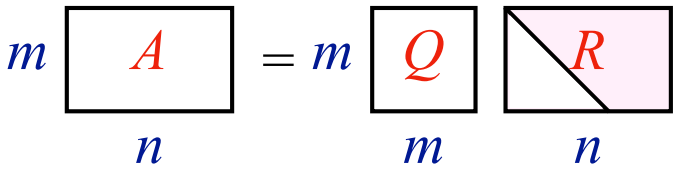
```
[[0.97722038 0.6754939 ]
 [0.24202867 0.08489658]
 [0.59948027 0.28712518]]
[[-0.85934042  0.49600353]
 [-0.18115481 -0.5230044 ]
 [-0.47824363 -0.69314277]]
[1.38100102 0.10812326]
[[-0.84743403 -0.53090072]
 [-0.53090072  0.84743403]]
```

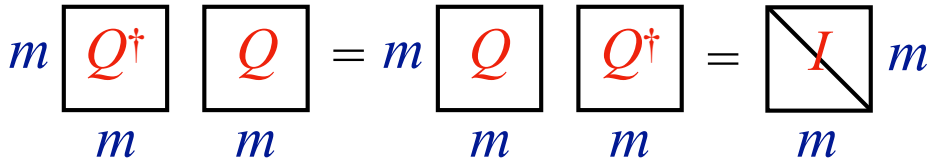
```
In [81]: print(np.dot(U,np.transpose(np.conj(U))))
         print(np.dot(np.transpose(np.conj(U)),U))
         print(np.dot(V,np.transpose(np.conj(V))))
         print(np.dot(np.transpose(np.conj(V)),V))
```

```
[[ 0.98448546 -0.10373837  0.06717283]
 [-0.10373837  0.30635067  0.44915286]
 [ 0.06717283  0.44915286  0.70916387]]
[[ 1.00000000e+00 -6.98401697e-18]
 [-6.98401697e-18  1.00000000e+00]]
[[1. 0.]
 [0. 1.]]
[[ 1.00000000e+00 -1.81922081e-17]
 [-1.81922081e-17  1.00000000e+00]]
```

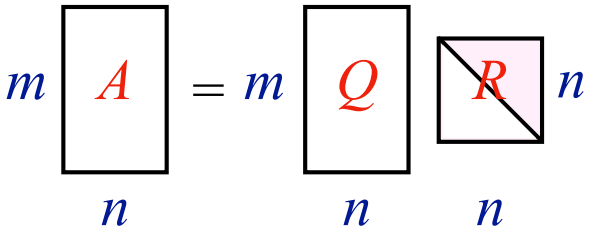
QR and LQ decomposition

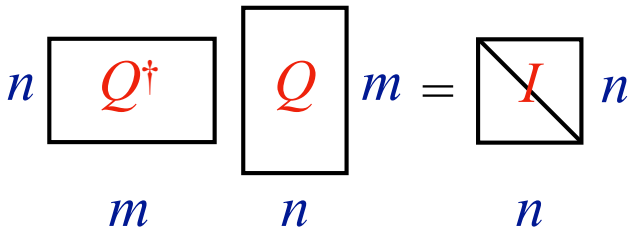
- QR decomposition
  - when $m < n$



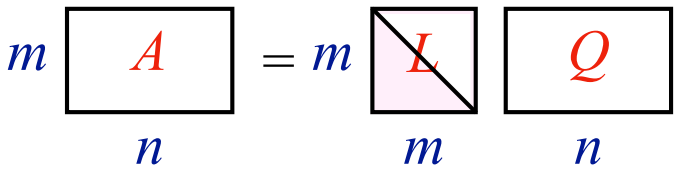


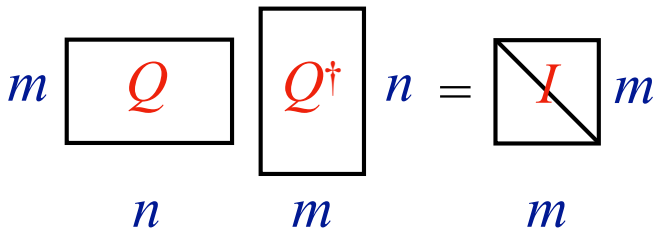
  - when $m > n$



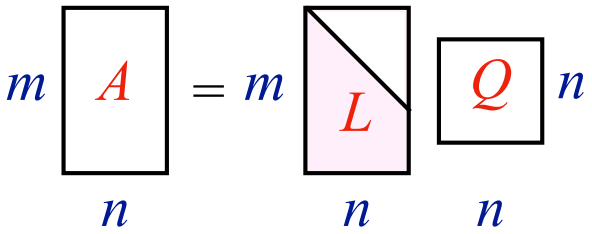


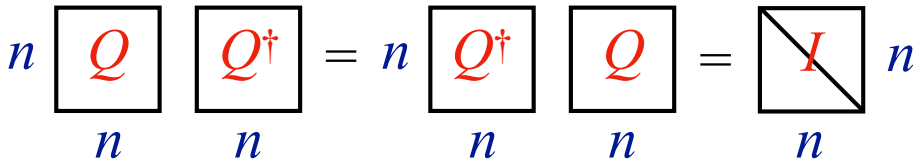
- LQ decomposition
  - when $m < n$





  - when $m > n$





Exercise

- review basic concepts of quantum mechanics
- 3-site Heisenberg model

$$H = \mathbf{S}_1 \cdot \mathbf{S}_2 + \mathbf{S}_2 \cdot \mathbf{S}_3$$

  find out all eigenvalues and eigenstates

- install python, e.g. Anaconda
  https://www.anaconda.com/download
- learn the basics of python, Numpy, Scipy, Matplotlib
  online tutorial: http://scipy-lectures.org/
  try matrix decompositions

Good Programming Habits

- plan and organize your code before you begin
- implement functions and classes
- use easy-to-read naming conventions throughout your code
- comment your code, even if it seems obvious
- optimize your code for efficiency through arrays, loops, etc
- test and debug your code as you build
- implement a system for version controls
- educate and expand your coding skill set