The quantum many-body non-equilibrium problem in phase space

Université de Strasbourg

cnrs

CESQ
CENTRE EUROPÉEN DE SCIENCES QUANTIQUES

www.schachenmayer.fr

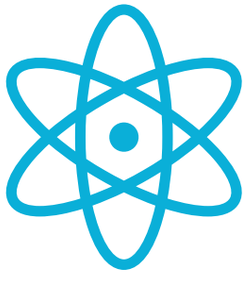
www.cesq.eu

Johannes Schachenmayer: **schachenmayer@unistra.fr**

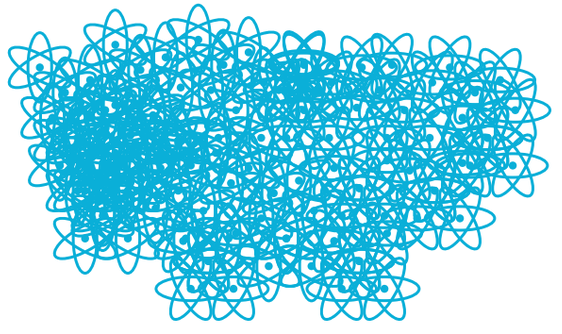
The “many-body quantum frontier”

Challenge: Controlled study of macroscopic coherent quantum superposition states

Microscopic world



Macroscopic world



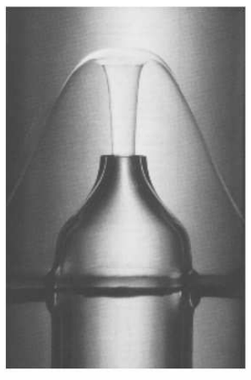
Quantum physics → *Classical physics*

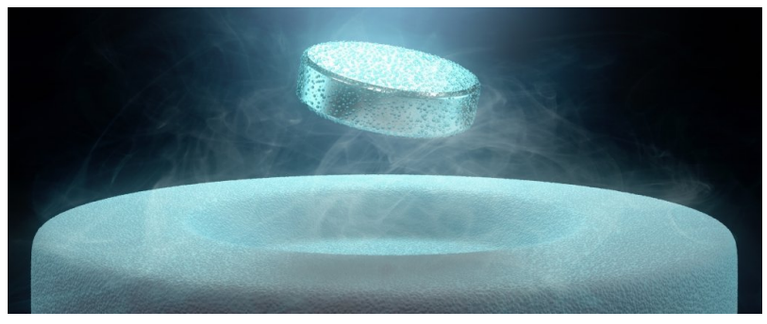
(decoherence)

$$\frac{d}{dt}|\psi\rangle = -\frac{\mathrm{i}}{\hbar}\hat{H}|\psi\rangle$$

$$\boldsymbol{F} = m\boldsymbol{a}$$

Macroscopic quantum effects





very rare, low temperatures

The “many-body quantum frontier”

Challenge: Controlled study of macroscopic coherent quantum superposition states

- **Many-body system:** *N particles* *Gigantic Hilbert space* $\dim(\mathcal{H}) \sim \exp(N)$

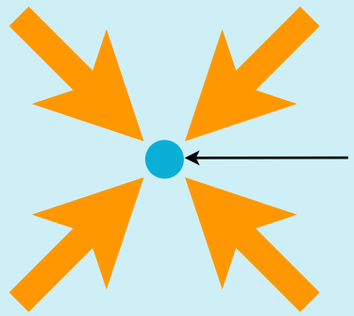
$\mathcal{H}$

Reasons:

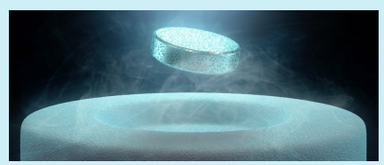
Noise

Disorder

Interaction range



Physics we know
(needs very few states)



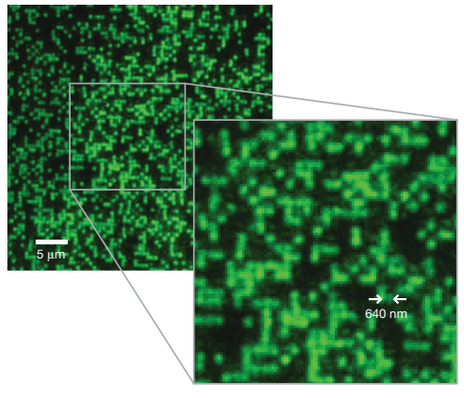
If instead we could create much larger superpositions controllably:

- **Fundamental question:** *Does quantum physics hold on the large scale?*
- **Applications:** *Engineering material properties, enhanced sensing, computing … ?*

The “many-body quantum frontier” - cold atom experiments

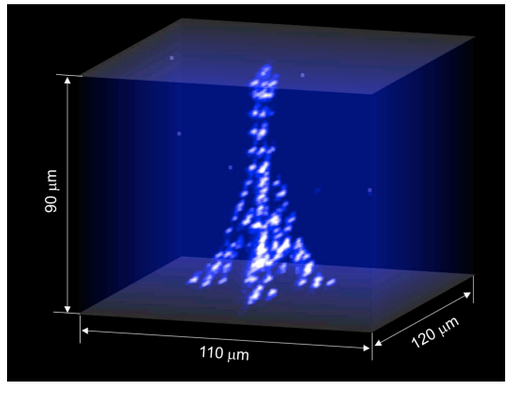
- Experimental platforms available with **cold atom physics:**

Optical lattices



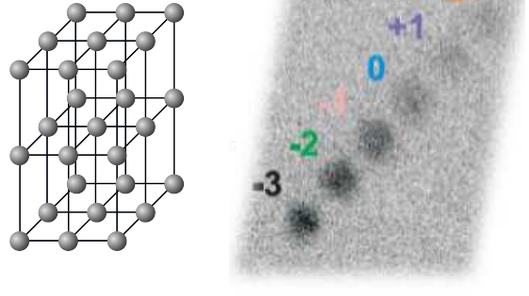
Munich, Harvard, Innsbruck, …
many more

Optical tweezers/Rydberg



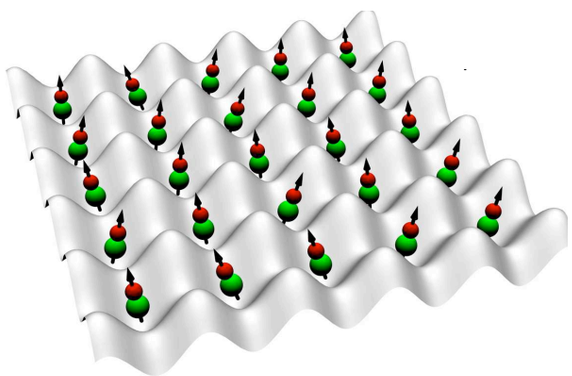
Paris, Harvard, … many more,
Strasbourg

Magnetic atoms

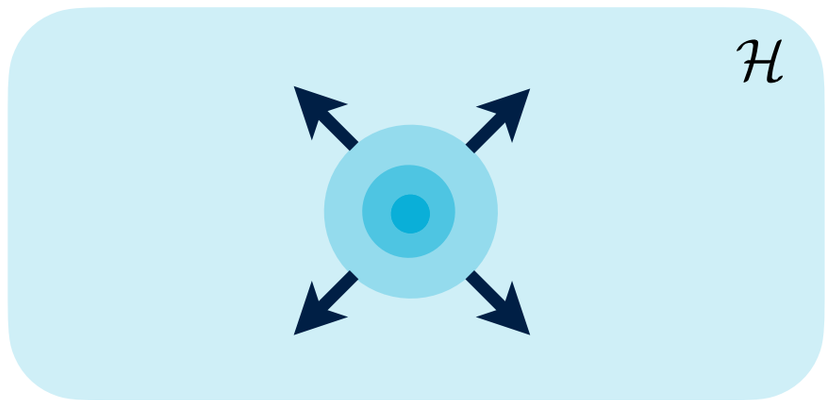


Innsbruck, Paris, Stuttgart …
many more

Polar molecules



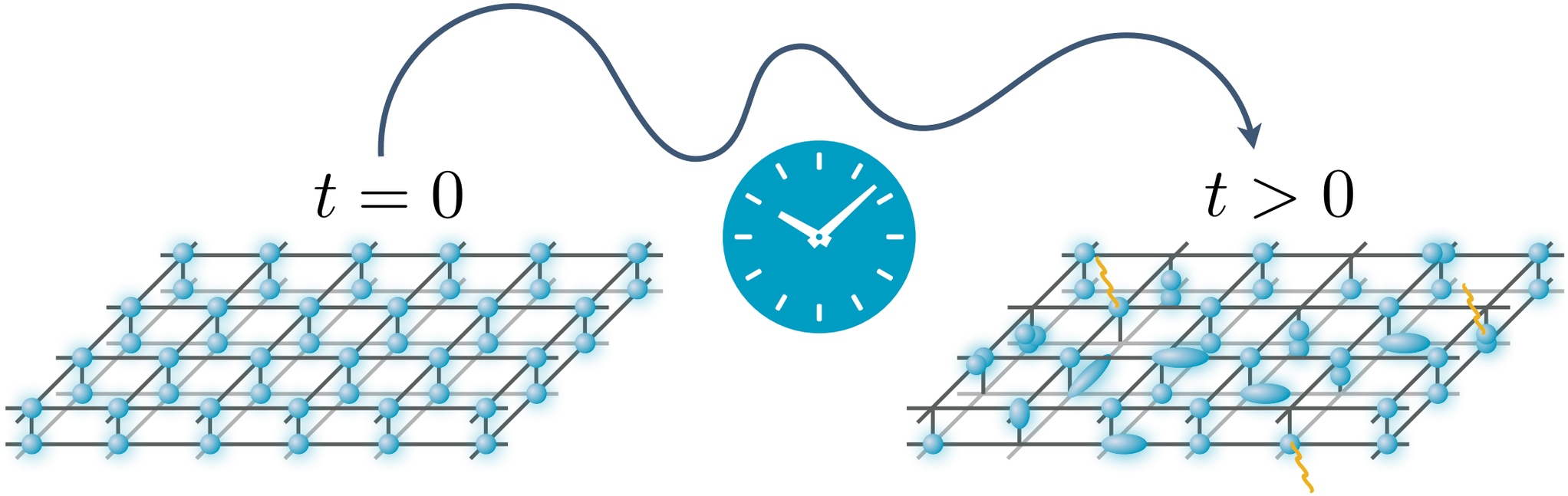
Boulder, Innsbruck, Ulm, …
many more



Large Hilbert space access (long coherence times)

…or any quantum computing platform

The “many-body quantum frontier” - non-equilibrium



$|\psi_0\rangle$ *…far out-of equilibrium*

$|\psi(t)\rangle$

$$\frac{d}{dt}|\psi\rangle = -\mathrm{i}\hat{H}|\psi\rangle$$

No decoherence

$$\frac{d}{dt}\hat{\rho} = -\mathrm{i}[\hat{H}, \hat{\rho}] + \sum_i \mathcal{L}^{[i]}\hat{\rho}$$

Master-equation noise modeling

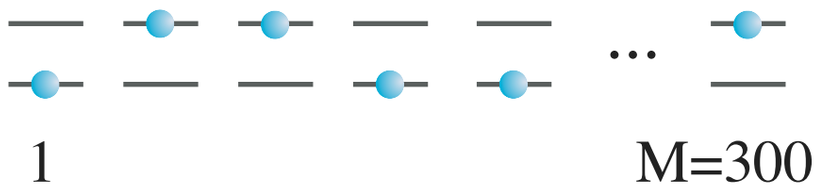
$\hbar \equiv 1$

Numerics: A fundamentally interesting problem

When do numerical simulations become impossible?

$|\psi\rangle \in \mathcal{H}$

$\dim(\mathcal{H}) = 2^M$



$\sim 10^{82}$ gigabyte

Atoms in universe (estimated): $\sim 10^{80}$

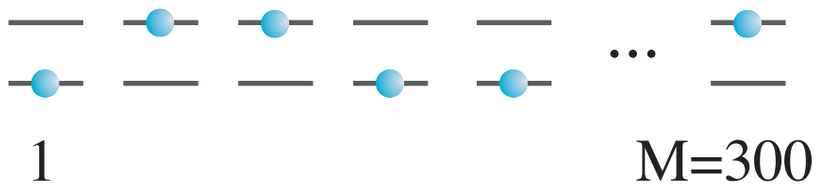
*So we need a **quantum computer** to simulate this?*

Numerics: A fundamentally interesting problem

When do numerical simulations become impossible?

$|\psi\rangle \in \mathcal{H}$

$\dim(\mathcal{H}) = 2^M$



$\sim 10^{82}$ gigabyte

Atoms in universe (estimated): $\sim 10^{80}$

We don't always need a quantum computer to simulate this!

Numerical simulations: A practical tool to understand quantum complexity better!

Numerics: A fundamentally interesting problem

Quantum many-body nonequilibrium dynamics

$$\frac{d}{dt}|\psi\rangle = -\mathrm{i}\hat{H}|\psi\rangle$$

$$\frac{d}{dt}\hat{\rho} = -\mathrm{i}[\hat{H},\hat{\rho}] + \sum_i \mathcal{L}^{[i]}\hat{\rho}$$

$$\hbar \equiv 1$$

Motivation:

- Model experiments & benchmark the status of quantum platforms (quantum advantage?)
- Find new emergent macroscopic phenomena
- Fundamentally understand quantum many-body from a classical complexity perspective

The quantum many-body non-equilibrium problem in phase space

Goal:

A tour through numerical methods for simulating large quantum many-body dynamics (in general and) with a focus on phase space concepts

Lecture 1: Exact numerical methods and their limitations. How to to go beyond?

Lecture 2: Introduction to quantum physics on phase space. The truncated Wigner approximation (TWA).

Lecture 3: Simulating quantum many-body dynamics with the discrete TWA (DTWA)

- Some text recommendations:

  - For phase-space methods: Online lecture notes: A. Polkovnikov, Boulder summer school 2013:
  https://boulderschool.yale.edu/2013/boulder-school-2013-lecture-notes

- Two types of lectures: Theory intro and "tutorial style" (sometimes mixed)
  Will provide code snippets, but no copy and paste!
- What these lectures are **not:**
  - Complete: Many techniques are not discussed, many proofs are skipped: The big picture instead
  - Computer science class
- Language recommendation (used for examples): Julia, **https://julialang.org/** (open source, easy, fast linear algebra)
- Philosophy: Use toolboxes only for basics (linear algebra, statistics)… let's understand things from scratch

$$h \equiv 1$$

… always!

$$\hbar \equiv 1$$

… always!

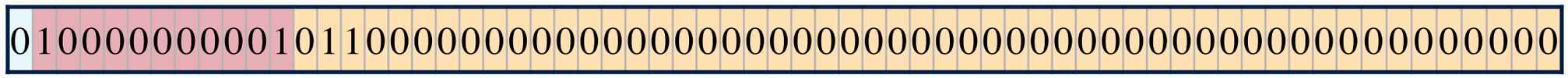
Semi-classical approximations $\hbar \to 0$

… will specify if $\hbar \neq 1$

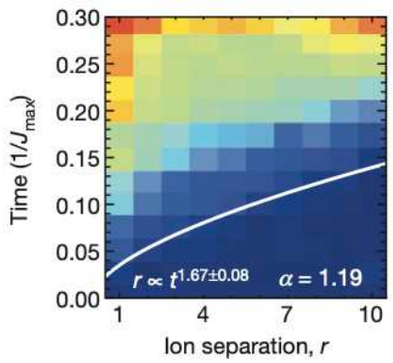
Lecture 1 - Plan for today

Today mostly: Tutorial style

- **Part 1.1:** Some fundamentals about numbers in digital memory and the linear algebra of quantum mechanics



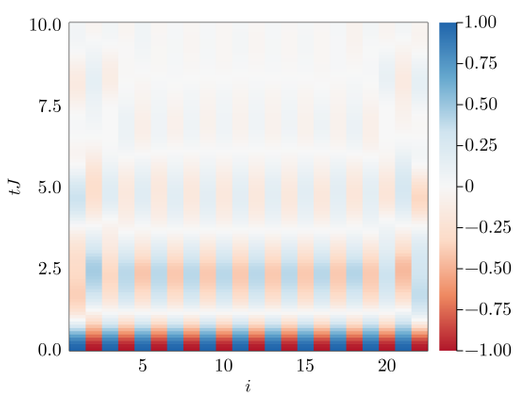
- **Part 1.2:** Constructing (sparse) Hamiltonians for many-body spin models



$$\cdots \otimes \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \otimes \cdots$$

spin #i *spin #j*

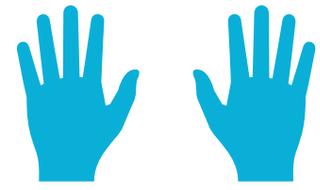
- **Part 1.3:** Krylov space methods. Applications to spin-model dynamics.



- **Part 1.4:** Quick discussion: How to go to larger systems?

Lecture 1.1 - Integers on a computer

Latin: digitus = finger



- Computers are digital, they work with bit representations of numbers
- Integers (signed, example with 4 bits)

$m=+0$	$m=+1$	$m=+2$	$m=+3$	$m=+4$	$m=+5$	$m=+6$	$m=+7$
0 0 0 0	0 0 0 1	0 0 1 0	0 0 1 1	0 1 0 0	0 1 0 1	0 1 1 0	0 1 1 1

Sign bit *Bit vector:* d_i $d_2\,d_1\,d_0$

$$m = d_2 2^2 + d_1 2^1 + d_0 2^0 = \sum_{i=0}^{2} d_i 2^i$$

$m=-0$	$m=-1$	$m=-2$	$m=-3$	$m=-4$	$m=-5$	$m=-6$	$m=-7$
1 0 0 0	1 0 0 1	1 0 1 0	1 0 1 1	1 1 0 0	1 1 0 1	1 1 1 0	1 1 1 1

- To not store the double zeros, usual convention is:

$m=-8$	$m=-7$	$m=-6$	$m=-5$	$m=-4$	$m=-3$	$m=-2$	$m=-1$
1 0 0 0	1 0 0 1	1 0 1 0	1 0 1 1	1 1 0 0	1 1 0 1	1 1 1 0	1 1 1 1

- Representable range of numbers (n bits): $(-2^{(n-1)}), \ldots, (2^{(n-1)} - 1) \sim -10^{18}, \ldots, 10^{18}$ *(n=64)*

```
julia> m = 1
```

```
julia> typeof(m)
Int64
```

```
julia> m = 2^63-1
9223372036854775807
```

```
julia> m + 1
-9223372036854775808
```

- **Tip:** When really pushing code performance it can sometimes help to use bitwise operations

Example: Figuring out qubit basis-state numbers

Lecture 1.1 - Floating point numbers on a computer

- Floating point numbers: $\pm 1.\text{fraction} \times 10^{\text{exponent}}$ … using **64 bits** (“double precision”)

$r =$ *Standard: IEEE-754*

$E_1 E_0 F_0 F_1 F_2$

`0 10000000001 0110000000000000000000000000000000000000000000000000`

11 bits *52 bits*

Sign bit (S) *Exponent ($\boldsymbol{E}$)* *Fraction ($\boldsymbol{F}$)*

Fractions should always be normalized, therefore we can gain a bit and store only

$$1.F$$

- Then, in binary represented numbers are

$$\textit{fraction:} \quad \mathcal{F} = 1 + F_0\, 2^{-1} + F_1\, 2^{-2} + F_2\, 2^{-3} + \cdots = 1 + \sum_{i=0}^{51} F_i\, 2^{-i-1}$$

$$\textit{exponent convention:} \quad \mathcal{E} = E - 1023 = \left(\sum_{i=0}^{10} E_i\, 2^i \right) - 1023$$

$$r = (-1)^S \times \mathcal{F} \times 2^{\mathcal{E}}$$

- **Example:** *(above)* $\mathcal{E} = 2^{10} + 2^0 - (2^{10} - 1) = 2$ $\quad \mathcal{F} = 2^0 + 2^{-2} + 2^{-3} = 1.375$

$$r = (-1)^0 \times 1.375 \times 2^2 = +5.5$$

- **Important:** Relative precision $1.0 \approx 1.0 \pm 2^{-53} \approx 1.0 \pm 10^{-16}$
- This means in practice: **Keep units normalized and consider everything below 1e-16 zero** $\hbar \equiv 1$ *definitely!*

In practice, pick a time unit by normalizing an energy, e.g.: $\hat{H} = -J \sum_i (\hat{\sigma}_i \hat{\sigma}_{i+1}^\dagger + \hat{\sigma}_i^\dagger \hat{\sigma}_{i+1})$ $\quad J \equiv 1$

time units $[1/J]$

Lecture 1.1 - Linear algebra of quantum mechanics

State = Vector

$$|\psi\rangle = \begin{bmatrix} \vdots \\ \psi_i \\ \vdots \end{bmatrix} \quad D \times 1$$

In general: Complex elements

$$D \times (2 \times 64)\,\text{Bits} = D \times 16\,\text{Bytes}$$

Operators = Matrix

$$\hat{H} = \begin{bmatrix} \vdots & & \vdots \\ & h_{i,j} & \\ \vdots & & \vdots \end{bmatrix} \quad D \times D$$

Hamiltonian, Observables, Time-evolution operator

- Let's define a state-vector as a general concept (not limited to linear quantum mechanics)

$$\mathbf{y}(t) \begin{bmatrix} \vdots \end{bmatrix}$$

Ordinary differential equation

$$\dot{\mathbf{y}}(t) = f(t, \mathbf{y}(t))$$

Schrödinger equation

$$\frac{d}{dt}|\psi\rangle = -\mathrm{i}\hat{H}|\psi\rangle$$

$$\mathbf{y} = |\psi\rangle \qquad f(t, \mathbf{y}(t)) = \hat{H}\,|\psi\rangle \qquad \textit{(linear)}$$

- The state-vector can be anything:

Linearized density matrix

$$\mathbf{y} = [\rho_{1,1}, \rho_{1,2}, \rho_{2,1}, \rho_{2,2}]^T$$

Two classical particles

$$\mathbf{y} = [x_1, p_1, x_2, p_2]^T$$

...

Lecture 1.1 - Linear algebra of quantum mechanics

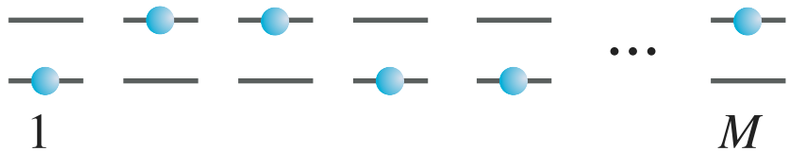
- Fundamental limit, how far can we go?

Double precision $D \times (2 \times 64)\,\text{Bits} = D \times 16\,\text{Bytes}$

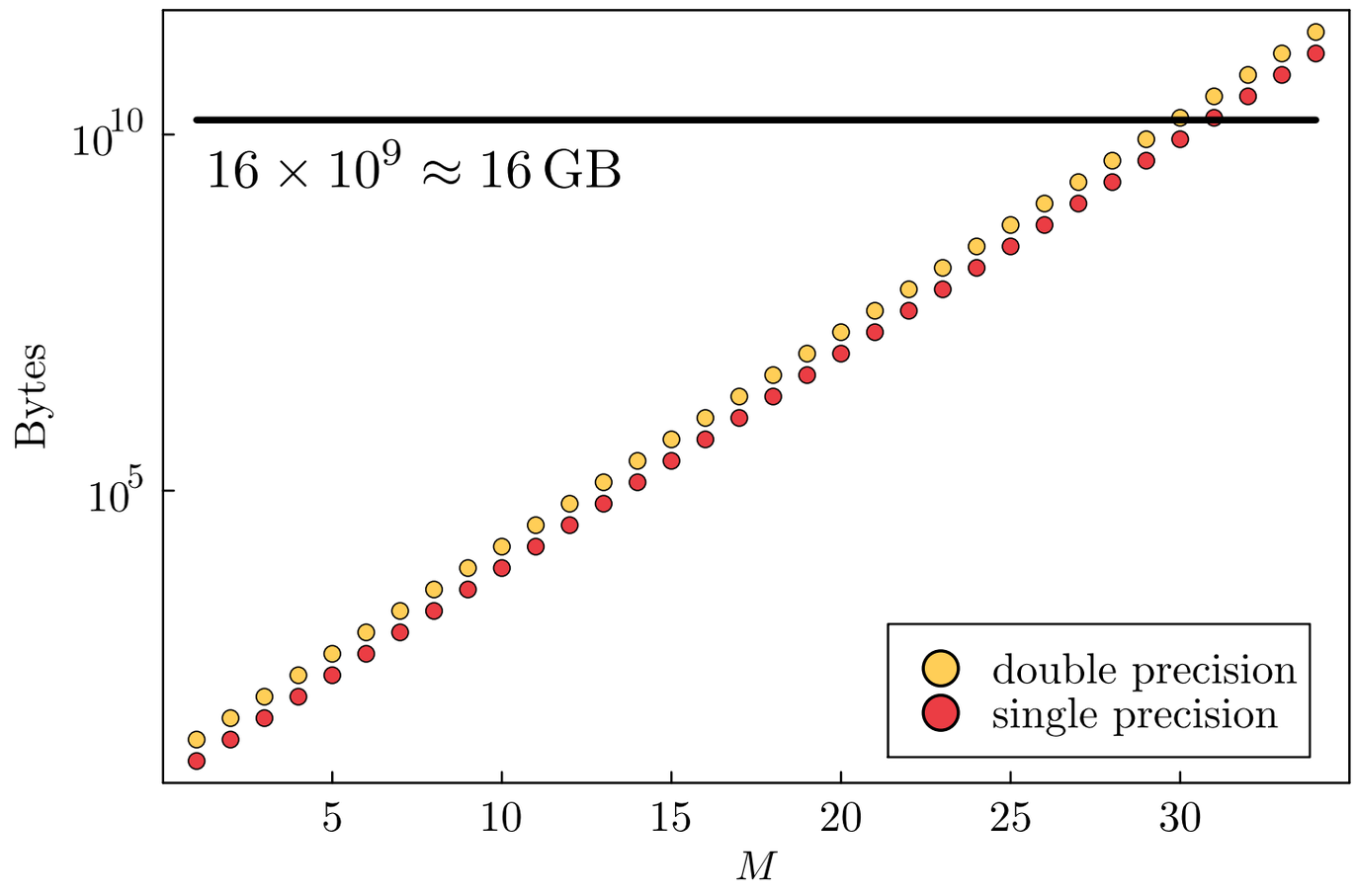
- Ultimately this will mean a memory limitation

Single precision $D \times 8\,\text{Bytes}$

- System of qubits



$$D = 2^M$$



$M \approx 30$

doable on current hardware

- **Remark:** Of course. a generic Hamiltonian would need memory $\sim D^2$, but we never need to store $\sim D^2$ elements when e.g. using Krylov space methods (see below)

- **Remark:** This is only the limit of exact state vector simulations, in practice with advanced methods, much larger systems are easily simulatable (see below)

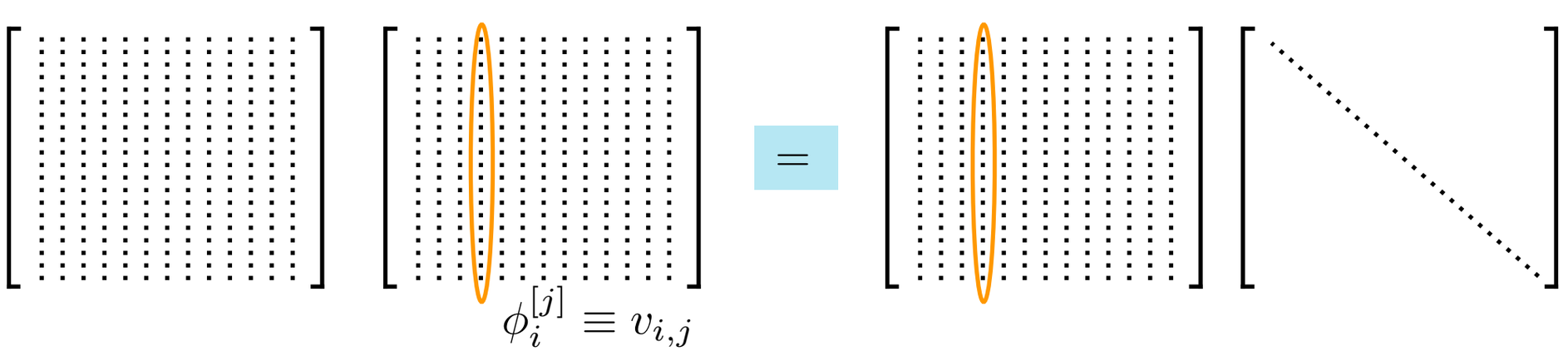
Lecture 1.1 - Exact Diagonalization

- Exact simulations of quantum mechanics often loosely described as: **Exact diagonalization (ED)**
- **Summary:** Diagonalization

Hamiltonian is Hermitian: real eigenvalues, unitary transformation $\hat{H}^\dagger = \hat{H}$ $D \times D$ *matrix*

Unitary matrix $\hat{V}^\dagger \hat{V} = \mathbb{1}$ $\hat{V}^\dagger = \hat{V}^{-1}$ $\hat{V}^\dagger \hat{H} \hat{V} = \hat{E}$ $\Leftrightarrow$ $\hat{H}\hat{V} = \hat{V}\hat{E}$ $\Leftrightarrow$ $\hat{H} = \hat{V}\hat{E}\hat{V}^\dagger$

$(\hat{H})_{i,j} \equiv h_{i,j}$ $(\hat{V})_{i,j} \equiv v_{i,j}$ $(\hat{V})_{i,j} \equiv v_{i,j}$ $(\hat{E})_{i,j} \equiv e_{i,j}$



$\phi_i^{[j]} \equiv v_{i,j}$

Diagonal matrix

$e_{i,j} \equiv \delta_{i,j} E_j$

- The j-th column of the matrix *v* is the eigenvector corresponding to the j-th eigenvalue:

$$\sum_k h_{i,k} v_{k,j} = \sum_k v_{i,k} e_{k,j} = E_j v_{i,j} \qquad \phi_i^{[j]} \equiv v_{i,j} \qquad \sum_k h_{i,k} \phi_k^{[j]} = E_j \phi_i^{[j]}$$

- The columns of V are the "eigenkets" $(|\phi_j\rangle)_i \equiv v_{i,j}$ $\hat{H}\,|\phi_j\rangle = E_j\,|\phi_j\rangle$
- … the rows are the "eigenbras" (complex conjugated eigenvectors), which follows from $\hat{V}^\dagger \hat{H} = \hat{E}\hat{V}^\dagger$

Lecture 1.1 - Exact Diagonalization

- Exact simulations of quantum mechanics often loosely described as: **Exact diagonalization (ED)**
- **Summary:** Diagonalization

Hamiltonian is Hermitian: real eigenvalues, unitary transformation $\hat{H}^\dagger = \hat{H}$ $D \times D$ *matrix*

Unitary matrix $\hat{V}^\dagger \hat{V} = \mathbb{1}$ $\hat{V}^\dagger = \hat{V}^{-1}$ $\hat{V}^\dagger \hat{H} \hat{V} = \hat{E}$ $\Leftrightarrow$ $\hat{H}\hat{V} = \hat{V}\hat{E}$ $\Leftrightarrow$ $\hat{H} = \hat{V}\hat{E}\hat{V}^\dagger$

$$(\hat{E})_{i,j} \equiv \delta_{i,j} E_j$$

- A diagonalization gives us a full spectral decomposition:

$$\hat{H} = \sum_k E_k \left|\phi_k\right\rangle \left\langle\phi_k\right|$$

- With this we can solve Schrödinger equation time-evolution

$$\frac{d}{dt}|\psi\rangle = -\mathrm{i}\hat{H}|\psi\rangle \qquad |\psi(t)\rangle = \mathrm{e}^{-\mathrm{i}t\hat{H}} |\psi(t=0)\rangle$$

$$\hat{V}^\dagger \hat{V} = \mathbb{1} \Leftrightarrow \langle\phi_k|\phi_l\rangle = \delta_{k,l}$$

$$\left(\left|\phi_k\right\rangle \left\langle\phi_k\right|\right)^n = \left|\phi_k\right\rangle \left\langle\phi_k\right|$$

Time-evolution operator (matrix exponential)

$$\mathrm{e}^{-\mathrm{i}\hat{H}t} = \sum_{n=0} \frac{(-\mathrm{i}t\hat{H})^n}{n!} = \sum_{n=0} \frac{(-\mathrm{i}t \sum_k E_k \left|\phi_k\right\rangle \left\langle\phi_k\right|)^n}{n!} = \sum_k \mathrm{e}^{-\mathrm{i}tE_k} \left|\phi_k\right\rangle \left\langle\phi_k\right|$$

- Diagonalization allows to compute the matrix exponential, and to compute exact evolution (no time-stepping needed)

$$|\psi(t)\rangle = \sum_k \mathrm{e}^{-\mathrm{i}tE_k} \left|\phi_k\right\rangle \left\langle\phi_k|\psi(t=0)\right\rangle$$

Lecture 1.1 - Exact Diagonalization

- Create random Hamiltonian

```
julia> H = rand(ComplexF64, 100, 100); H += H'
```

$$\hat{V}^\dagger \hat{H} \hat{V} = \hat{E} \quad \Leftrightarrow \quad \hat{H}\hat{V} = \hat{V}\hat{E} \quad \Leftrightarrow \quad \hat{H} = \hat{V}\hat{E}\hat{V}^\dagger$$

- Compute V and E

```
julia> using LinearAlgebra

julia> E, V = eigen(H);
```

E comes out as vector

- Check equations:

```
julia> norm(H*V .- V*Diagonal(E))
2.3421977872625636e-13
```

```
julia> norm(H .- V*Diagonal(E)*V')
7.79771749285727e-13
```

Precision of diagonalization algorithm (not quite double precision, but enough)

- Time evolution with matrix exponential

$$|\psi(t)\rangle = \sum_k \mathrm{e}^{-\mathrm{i}tE_k} \, |E_k\rangle \, \langle E_k|\psi(t=0)\rangle$$

```
psi0 = zeros(ComplexF64, D)
psi0[1] = 1.0
psit = zeros(ComplexF64, D)
for kk = 1:D
    ovl = V[:, kk]' * psi0
    psit += exp(-1im * t * E[kk]) * ovl .* V[:, kk]
end
```

- Also possible: Build matrix exponential directly with built-in routines:

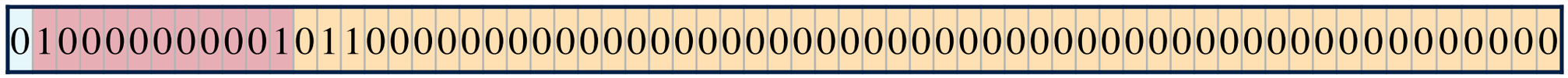
```
julia> U = exp(-1im .* t .* H); psit = U*psi0
100-element Vector{ComplexF64}:
```

Warning: Make sure exp is not element-wise exponential!

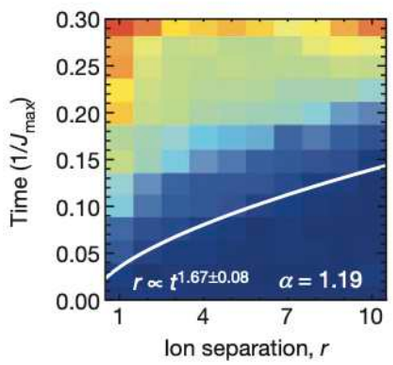
Lecture 1.1 - Plan for today

Today mostly: Tutorial style

- **Part 1.1:** Some fundamentals about numbers in digital memory and the linear algebra of quantum mechanics



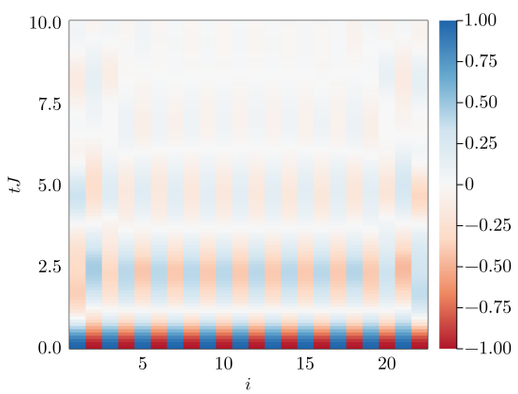
- **Part 1.2:** Constructing (sparse) Hamiltonians for many-body spin models



$$\cdots \otimes \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \otimes \cdots$$

spin #i *spin #j*

- **Part 1.3:** Krylov space methods. Applications to spin-model dynamics



- **Part 1.4:** Quick discussion: How to go to larger systems?

Lecture 1.2 - Motivation: Spin-model quench dynamics

- Experiments implementing long-range spin-models effectively (e.g. with trapped ions)

$$J_{ij} = \frac{J}{|i-j|^{\alpha}}$$

D. Porras & I. J. Cirac, Phys. Rev. Lett. 92, 207901 (2004)

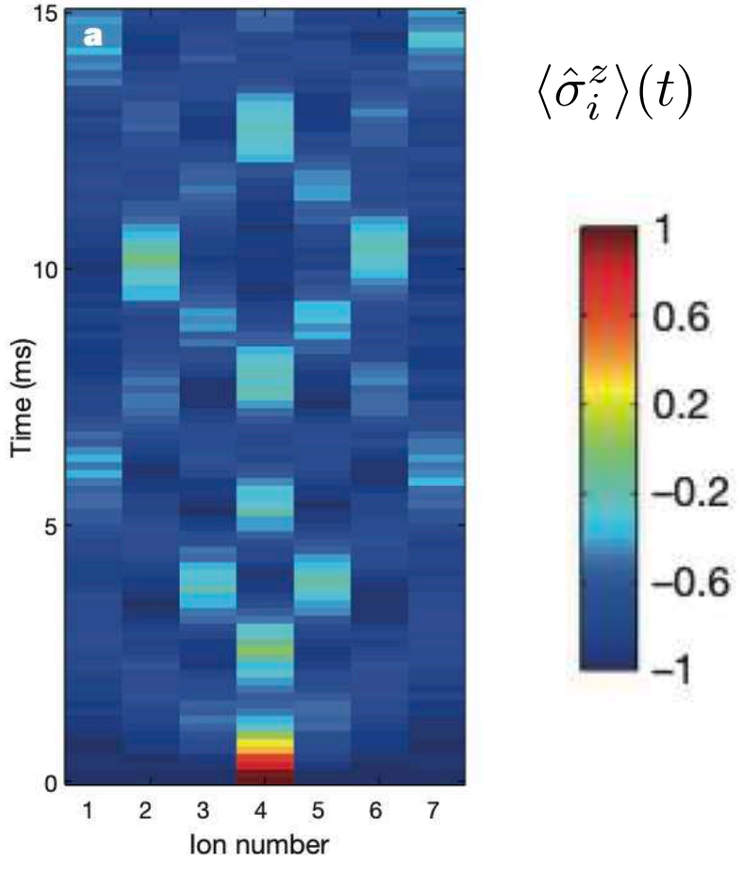
$$\hat{H}_{\mathrm{TI}} = \sum_{i<j} J_{i,j}\hat{\sigma}_i^z\hat{\sigma}_j^z + h_x\sum_i \hat{\sigma}_i^x$$

Long-range transverse Ising model

P. Jurcevic, et al. Nature 511, 202 (2014)
Innsbruck

$$|\psi(t=0)\rangle = |\downarrow\ldots\downarrow\uparrow\downarrow\ldots\downarrow\rangle$$

$$\langle\hat{\sigma}_i^z\rangle(t)$$



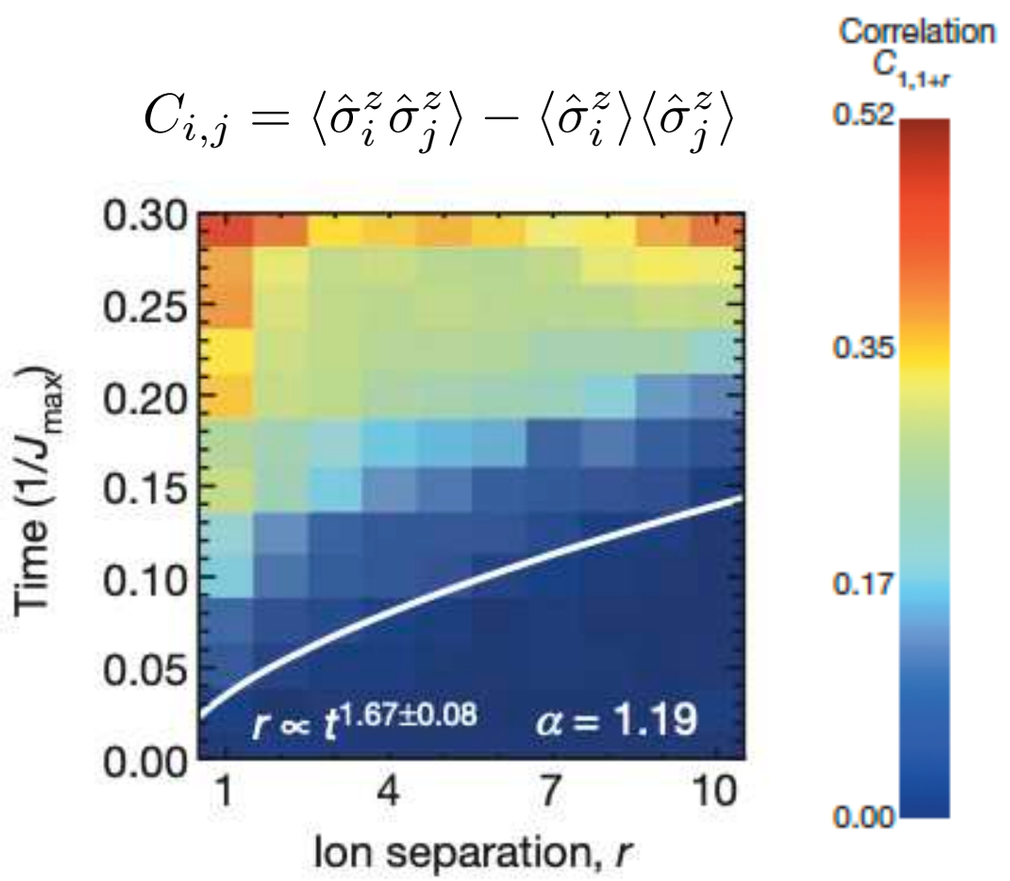
$$\hat{H}_{\mathrm{XY}} = \sum_{i<j} J_{i,j}\left(\hat{\sigma}_i^x\hat{\sigma}_j^x + \hat{\sigma}_i^y\hat{\sigma}_j^y\right) = 2\sum_{i<j} J_{i,j}\left(\hat{\sigma}_i^+\hat{\sigma}_j^- + \hat{\sigma}_i^-\hat{\sigma}_j^+\right)$$

Long-range XY model

P. Richerme, et al. Nature 511, 198 (2014)
JQI Maryland

$$|\psi(t=0)\rangle = |\downarrow\downarrow\ldots\downarrow\rangle$$

$$C_{i,j} = \langle\hat{\sigma}_i^z\hat{\sigma}_j^z\rangle - \langle\hat{\sigma}_i^z\rangle\langle\hat{\sigma}_j^z\rangle$$



Lecture 1.2 - Hilbert space construction

- In a many-body system, how do we construct Hamiltonian matrices for such spin model?

$$\hat{H}_{\mathrm{TI}} = \sum_{i<j} J_{i,j}\hat{\sigma}_i^z\hat{\sigma}_j^z + h_x \sum_i \hat{\sigma}_i^x \qquad \textit{Transverse Ising model}$$

- What Hamiltonian terms actually mean: All terms are matrices: $2^N \times 2^N$

$$\hat{\sigma}_i^x = \cdots \otimes \mathbb{1} \otimes \hat{\sigma}^x \otimes \mathbb{1} \otimes \cdots = \cdots \otimes \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \otimes \cdots$$

spin #i *spin #i*

"tensor product" = "Kronecker product"

$$\hat{\sigma}_i^z\hat{\sigma}_j^z = \cdots \otimes \mathbb{1} \otimes \hat{\sigma}^z \otimes \mathbb{1} \otimes \mathbb{1} \otimes \hat{\sigma}^z \otimes \mathbb{1} \otimes \cdots =$$

$$\cdots \otimes \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \otimes \cdots$$

spin #i *spin #j*

- Then, e.g. easy to construct a spin-lowering for spin *i:*

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

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

$$\hat{\sigma}^- = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$$

$$\hat{\sigma}_i^- = \underset{2^{i-1}\times 2^{i-1}}{\begin{pmatrix} 1 & & \\ & 1 & \\ & & \ddots \end{pmatrix}} \otimes \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \otimes \underset{2^{N-i}\times 2^{N-i}}{\begin{pmatrix} 1 & & \\ & 1 & \\ & & \ddots \end{pmatrix}}$$

Lecture 1.2 - Hilbert space construction

$$\hat{H}_{\mathrm{TI}} = \sum_{i<j} J_{i,j} \hat{\sigma}_i^z \hat{\sigma}_j^z + h_x \sum_i \hat{\sigma}_i^x \qquad \textit{Transverse Ising model}$$

- Hamiltonian can be fully constructed from spin-lowering operators only

$$|\downarrow\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad |\uparrow\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad \hat{\sigma}^- = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$$

$$\hat{\sigma}_i^- = \underbrace{\begin{pmatrix} 1 & & \\ & 1 & \\ & & \ddots \end{pmatrix}}_{2^{i-1} \times 2^{i-1}} \otimes \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \otimes \underbrace{\begin{pmatrix} 1 & & \\ & 1 & \\ & & \ddots \end{pmatrix}}_{2^{N-i} \times 2^{N-i}}$$

```
# Construct lowering operators
N = 10
sm = [0 1; 0 0]
sms = Vector{}(undef, N)

for ii = 1:N
    ldim = 2^(ii-1)
    rdim = 2^(N - ii)
    sms[ii] = kron(Matrix(I, ldim, ldim), sm, Matrix(I, rdim, rdim))
end
```

Vector of lowering operators, all that's needed!

Lecture 1.2 - Hilbert space construction

$$\hat{H}_{\mathrm{TI}} = \sum_{i<j} J_{i,j}\hat{\sigma}_i^z\hat{\sigma}_j^z + h_x \sum_i \hat{\sigma}_i^x \qquad \textit{Transverse Ising model}$$

- Then, full Hamiltonian construction

```
# Construct Hamiltonian
Jij = rand(N,N) # random couplings
hx = 1;

H = zeros(2^N, 2^N)
for ii = 1:N
    H += hx .* (sms[ii]' + sms[ii]) # sx
    szii = (sms[ii]'*sms[ii] - sms[ii]*sms[ii]') # sz spin ii
    for jj = (ii+1):N
        szjj = (sms[jj]'*sms[jj] - sms[jj]*sms[jj]') # sz spin jj
        H += Jij[ii,jj] .* szii * szjj
    end
end
```

$$\hat{\sigma}_i^x = \hat{\sigma}_i^- + \hat{\sigma}_i^+$$

$$\hat{\sigma}_i^z = \hat{\sigma}_i^+\hat{\sigma}_i^- - \hat{\sigma}_i^-\hat{\sigma}_i^+$$

- **Problem:** Not memory efficient!

Lecture 1.2 - Sparse Matrices!

$$\hat{H}_{\mathrm{TI}} = \sum_{i<j} J_{i,j} \hat{\sigma}_i^z \hat{\sigma}_j^z + h_x \sum_i \hat{\sigma}_i^x \qquad \textit{Transverse Ising model}$$

$$N = 10$$

- **Problem:** Not memory efficient!
- Even for a full coupling matrix, the Hamiltonian is very sparse!

  Most elements are zero!

- Elements in matrix: 2^{2N}
- Non-zero elements: $\mathcal{O}(2^N)$

```
1024×1024 Matrix{Float64}:
 21.607   1.0      1.0      0.0      …  0.0      0.0      0.0      0.0
  1.0    11.6526   0.0      1.0         0.0      0.0      0.0      0.0
  1.0     0.0     14.9075   1.0         0.0      0.0      0.0      0.0
  0.0     1.0      1.0      5.81673     0.0      0.0      0.0      0.0
  1.0     0.0      0.0      0.0         0.0      0.0      0.0      0.0
  0.0     1.0      0.0      0.0      …  0.0      0.0      0.0      0.0
  0.0     0.0      1.0      0.0         0.0      0.0      0.0      0.0
  0.0     0.0      0.0      1.0         0.0      0.0      0.0      0.0
  1.0     0.0      0.0      0.0         0.0      0.0      0.0      0.0
  0.0     1.0      0.0      0.0         0.0      0.0      0.0      0.0
  ⋮                                  ⋱  ⋮
  0.0     0.0      0.0      0.0         0.0      0.0      1.0      0.0
  0.0     0.0      0.0      0.0      …  0.0      0.0      0.0      1.0
  0.0     0.0      0.0      0.0         1.0      0.0      0.0      0.0
  0.0     0.0      0.0      0.0         0.0      1.0      0.0      0.0
  0.0     0.0      0.0      0.0         0.0      0.0      1.0      0.0
  0.0     0.0      0.0      0.0         0.0      0.0      0.0      1.0
  0.0     0.0      0.0      0.0      …  5.81673  1.0      1.0      0.0
  0.0     0.0      0.0      0.0         1.0      14.9075  0.0      1.0
  0.0     0.0      0.0      0.0         1.0      0.0      11.6526  1.0
  0.0     0.0      0.0      0.0         0.0      1.0      1.0      21.607
```

This example:

```
julia> sum((H .> 1e-16))
10698
```

```
julia> sum((H .> 1e-16)) / 2^20
0.010202407836914062
```

Only 1% filled … let's use **Sparse Matrices!**

- We should be using **sparse matrices**
  = a data format where we only store the non-zero elements (with their index positions)

Lecture 1.2 - Sparse Matrices!

- This can be done with minimal modifications

```
# Construct lowering operators
using SparseArrays
N = 10
sm = sparse([0 1; 0 0])
sms = Vector{}(undef, N)

for ii = 1:N
    ldim = 2^(ii-1)
    rdim = 2^(N - ii)
    sms[ii] = kron(sparse(I, ldim, ldim), sm, sparse(I, rdim, rdim))
end
```

```
# Construct Hamiltonian
using SparseArrays
Jij = rand(N,N) # random couplings
hx = 1;

H = spzeros(2^N, 2^N)
for ii = 1:N
    H += hx .* (sms[ii]' + sms[ii]) # sx
    szii = (sms[ii]'*sms[ii] - sms[ii]*sms[ii]') # sz spin ii
    for jj = (ii+1):N
        szjj = (sms[jj]'*sms[jj] - sms[jj]*sms[jj]') # sz spin jj
        H += Jij[ii,jj] .* szii * szjj
    end
end
```

Full vs. sparse version

```
61.911068 seconds
 0.004216 seconds
```

Lecture 1.2 - Hilbert space construction in a many-body system

- **Remark:** Initial product states can also easily produced with Kronecker products

$$|\psi_0\rangle = |\downarrow\uparrow\downarrow\uparrow\rangle = |\downarrow\rangle \otimes |\uparrow\rangle \otimes |\downarrow\rangle \otimes |\uparrow\rangle \quad \Leftrightarrow$$

```
spd = [1; 0]
spu = [0; 1]
psi0 = kron(spd,spu,spd,spu)
```

…or even more simply: $|\psi_0\rangle = |\downarrow\downarrow\downarrow\downarrow\rangle \quad \Leftrightarrow$

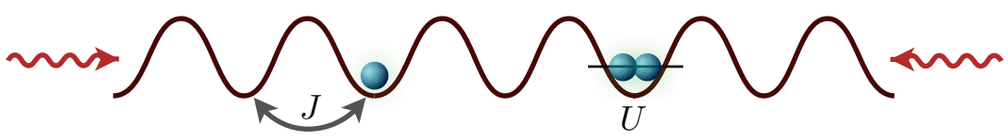
```
psi0 = zeros(2^N)
psi0[1] = 1.0
```

- **Remark:** We want to compute expectation values, this can e.g. again be easily achieved using sparse operators

$$\langle\psi(t)|\,\hat{\sigma}_i^z\,|\psi(t)\rangle \quad \Leftrightarrow$$

```
szjj = (sms[jj]'*sms[jj] - sms[jj]*sms[jj]') # sz spin jj
expc_szjj = real(psi'*szjj*psi)
```

- **Remark:** For a system of bosons in a lattice, very similar construction possible



$$\hat{H} = -J\sum_i(\hat{b}_i\hat{b}_{i+1}^\dagger + \hat{b}_i^\dagger\hat{b}_{i+1}) + \frac{U}{2}\sum_i \hat{b}_i^\dagger\hat{b}_i^\dagger\hat{b}_i\hat{b}_i$$

But: *Number not conserved!*

$$D = n_{\max}^N$$

Introduce cutoff of Fock space basis on each site

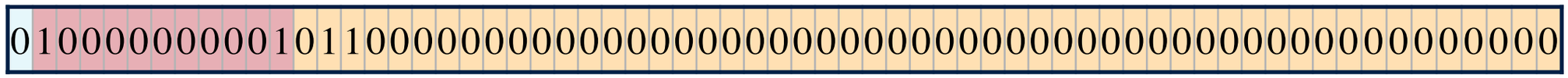
Define bosonic field operators and use Kroneckers as before

```
# |n=0> = [1;0;0;0; ...]
# |n=2> = [0;0;1;0; ...]
# b |n=2> = sqrt(n) * [0;1;0;0; ...]
nm = 5 # cutoff for max. nm-1 bosons)
b = sparse(1:(nm-1), 2:nm, sqrt.(1:(nm-1)), nm, nm)
```

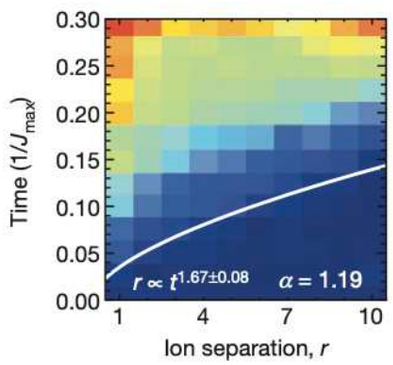
Lecture 1 - Plan for today

Today mostly: Tutorial style

- **Part 1.1:** Some fundamentals about numbers in digital memory and the linear algebra of quantum mechanics



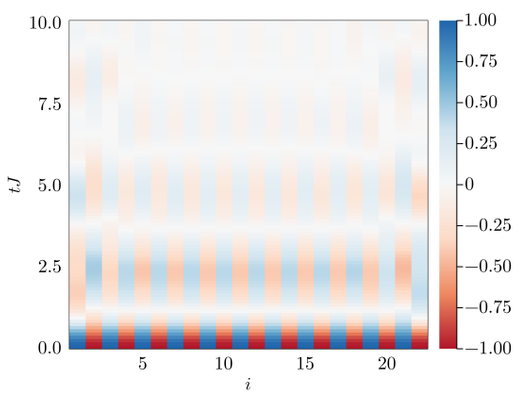
- **Part 1.2:** Constructing (sparse) Hamiltonians for many-body spin models



$$\cdots \otimes \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \otimes \cdots$$

spin #i *spin #j*

- **Part 1.3:** Krylov space methods. Applications to spin-model dynamics



- **Part 1.4:** Quick discussion: How to go to larger systems?

Lecture 1.3 - Krylov space methods

- The goal is to compute dynamics of some initial state

$$\frac{d}{dt}|\psi\rangle = -\mathrm{i}\hat{H}|\psi\rangle \qquad |\psi(t)\rangle = \mathrm{e}^{-\mathrm{i}\hat{H}t}\,|\psi_0\rangle \qquad \ldots \textit{with a generally very } \textbf{\textit{sparse matrix}}$$

- We could just use standard ODE solvers (e.g. Runge-Kutta, see lecture 3), but it would be overkill, since:

i) The problem is linear
ii) The Hamiltonian is time-independent

- A more efficient method can be to compute the matrix exponential directly

- **Problem:** A full matrix exponentiation turns the sparse Hamiltonian into a full matrix

E.g.: Compute matrix exponential by full diagonalization

$$\hat{H}\,|E_n\rangle = E_n\,|E_n\rangle \qquad \hat{H} = \sum_n E_n\,|E_n\rangle\langle E_n| \qquad \mathrm{e}^{-\mathrm{i}\hat{H}t} = \sum_{k=0}\frac{(-\mathrm{i}t\hat{H})^k}{k!} = \sum_n \mathrm{e}^{-\mathrm{i}tE_n}\,|E_n\rangle\langle E_n|$$

sparse ~D elements — *D energy eigenstates* — *each ~D elements* — *in general: Full DxD matrix … not sparse :(*

- **On the other hand:** We're only interested in the state after applying the matrix exponential

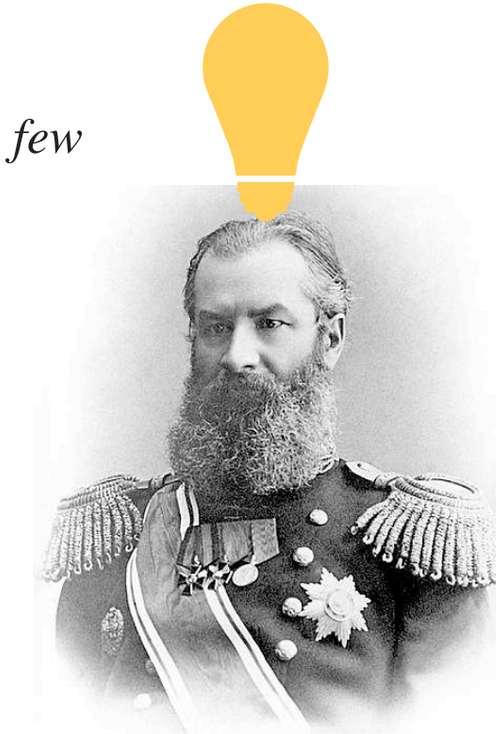
$$\mathrm{e}^{-\mathrm{i}\hat{H}t}\,|\psi_0\rangle = \sum_{k=0}\frac{(-\mathrm{i}t\hat{H})^k}{k!}\,|\psi_0\rangle \equiv \sum_{k=0}\frac{\hat{A}^k}{k!}\,|\psi_0\rangle$$

Maybe we only need a few terms in this sum?

Lecture 1.3 - Krylov space methods

$$e^{-i\hat{H}t}\left|\psi_0\right\rangle = \sum_{k=0} \frac{(-it\hat{H})^k}{k!}\left|\psi_0\right\rangle \equiv \sum_{k=0} \frac{\hat{A}^k}{k!}\left|\psi_0\right\rangle$$

Maybe we only need a few terms in this sum?



Aleksey Nikolaevich Krylov (1863 - 1945)

- Starting from the initial state, a good idea seems to be using the following space:

- **Krylov space:** $\text{span}\left(\hat{A}^0\left|\psi_0\right\rangle, \hat{A}^1\left|\psi_0\right\rangle, \hat{A}^2\left|\psi_0\right\rangle, \ldots, \hat{A}^{m-1}\left|\psi_0\right\rangle\right)$

(here formulated in bra/ket notation, but concept is general)

Krylov wrote about 300 papers and books. They span a wide range of topics, including shipbuilding, magnetism, artillery, mathematics, astronomy, and geodesy.

- **Problem:** The states are not orthogonal (in fact with increasing k they become more and more parallel)

Example, let's for a sec take the matrix to be a hermitian Hamiltonian $\hat{A} = \hat{H}$

Take an initial state expanded in eigenstates:

$$\left|\psi_0\right\rangle = \sum_n c_n \left|E_n\right\rangle \qquad \hat{H}\left|E_n\right\rangle = E_n\left|E_n\right\rangle \qquad E_1 > E_2 > E_3 > \ldots$$

k-th Krylov state $\left|\psi_k\right\rangle \equiv \hat{H}^k\left|\psi_0\right\rangle = \sum_n c_n E_n^k \left|E_n\right\rangle$

All high k states will be parallel to $\left|E_1\right\rangle$

- When normalizing and $k \gg 1$

$$c_1 E_1^k \gg c_2 E_2^k$$

$$\frac{\left|\psi_k\right\rangle}{\|\left|\psi_k\right\rangle\|} \approx \left|E_1\right\rangle$$

Also: *Repeated application gives the highest energy state! Known as* ***power method***

…assuming $c_1 \neq 0$

Lecture 1.3 - Krylov space methods - Arnoldi iteration

- **Krylov space:** $\text{span}\left(\hat{A}^0 |\psi_0\rangle, \hat{A}^1 |\psi_0\rangle, \hat{A}^2 |\psi_0\rangle, \ldots, \hat{A}^{m-1} |\psi_0\rangle\right)$

  (here formulated in bra/ket notation, but concept is general)

- **Problem:** The states are not orthogonal (in fact with increasing *k* they become more and more parallel)

- **Arnoldi iteration:** Create an orthonormal basis from the Krylov vectors (using improved/modified Gram-Schmidt)

$|\psi_0\rangle = |\psi_0\rangle / \| |\psi_0\rangle \|$ *(normalization just in case)*

1st iteration

$$|\psi_1\rangle \equiv \hat{A} |\psi_0\rangle$$
$$h_{1,1} \equiv \langle\psi_0|\psi_1\rangle \quad |\psi_1\rangle \equiv |\psi_1\rangle - h_{1,1} |\psi_0\rangle$$
$$h_{2,1} \equiv \| |\psi_1\rangle \| \quad |\psi_1\rangle \equiv |\psi_1\rangle / h_{2,1}$$

ith - iteration

$$|\psi_i\rangle \equiv \hat{A} |\psi_{i-1}\rangle$$
$$h_{1,i} \equiv \langle\psi_0|\psi_i\rangle \quad |\psi_i\rangle \equiv |\psi_i\rangle - h_{1,i} |\psi_0\rangle$$
$$h_{2,i} \equiv \langle\psi_1|\psi_i\rangle \overset{\bigstar}{} \quad |\psi_i\rangle \equiv |\psi_i\rangle - h_{2,i} |\psi_1\rangle$$
$$\ldots$$
$$h_{i,i} \equiv \langle\psi_{i-1}|\psi_i\rangle \quad |\psi_i\rangle \equiv |\psi_i\rangle - h_{i+1,i} |\psi_{i-1}\rangle$$
$$h_{i+1,1} \equiv \| |\psi_i\rangle \| \quad |\psi_i\rangle \equiv |\psi_i\rangle / h_{i+1,1}$$

Walter Edwin Arnoldi
(American engineer/mathematician)
(1917 - 1995)

Notation: $\equiv$

means "define"

…as in code:

```
= # not ==
```

★ *modified GS:*
use already updated state

Lecture 1.3 - Krylov space methods - Arnoldi iteration

- **Krylov space:** $\text{span}\left(\hat{A}^0|\psi_0\rangle, \hat{A}^1|\psi_0\rangle, \hat{A}^2|\psi_0\rangle, \ldots, \hat{A}^{m-1}|\psi_0\rangle\right)$

$$|\psi_i\rangle \equiv \hat{A}|\psi_{i-1}\rangle \qquad \textit{ith - iteration}$$

$$h_{1,i} \equiv \langle\psi_0|\psi_i\rangle \quad |\psi_i\rangle \equiv |\psi_i\rangle - h_{1,i}|\psi_0\rangle$$

$$h_{2,i} \equiv \langle\psi_1|\psi_i\rangle \quad |\psi_i\rangle \equiv |\psi_i\rangle - h_{2,i}|\psi_1\rangle$$

$$\ldots$$

$$h_{i,i} \equiv \langle\psi_{i-1}|\psi_i\rangle \quad |\psi_i\rangle \equiv |\psi_i\rangle - h_{i+1,i}|\psi_{i-1}\rangle$$

$$h_{i+1,1} \equiv \| |\psi_i\rangle \| \quad |\psi_i\rangle \equiv |\psi_i\rangle / h_{i+1,1}$$

after i iterations, we have i+1 orthogonal vectors, i.e. we have the "semi-unitary" matrix

$$\mathbf{Q}_i = \begin{pmatrix} |\psi_0\rangle & |\psi_1\rangle & \ldots & |\psi_i\rangle \end{pmatrix}$$

$$D \times (i+1) \qquad \mathbf{Q}_i^\dagger \mathbf{Q}_i = \mathbb{1}$$

- $\mathbf{Q}_i$ is a projection matrix to the Krylov basis
- What is the matrix **h?**

$$\mathbf{h}_i = \begin{pmatrix} h_{1,1} & h_{1,2} & \ldots & h_{1,i} \\ h_{2,1} & h_{2,2} & \ldots & h_{2,i} \\ 0 & h_{3,2} & \ldots & h_{3,i} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & h_{i+1,i} \end{pmatrix}$$

$$(i+1) \times i$$

Upper Hessenberg form

The iteration can be written in the matrix form:

$$\hat{A}\mathbf{Q}_i = \mathbf{Q}_{i+1}\mathbf{h}_i \qquad \textit{(exercise: show)}$$

Example: i=1

$$|\psi_1\rangle = \left(\hat{A}|\psi_0\rangle - h_{1,1}|\psi_0\rangle\right)/h_{2,1}$$

$$\hat{A}|\psi_0\rangle = h_{1,1}|\psi_0\rangle + h_{2,1}|\psi_1\rangle$$

$$\hat{A}(|\psi_0\rangle) = \begin{pmatrix} |\psi_0\rangle & |\psi_1\rangle \end{pmatrix} \begin{pmatrix} h_{1,1} \\ h_{2,1} \end{pmatrix}$$

Lecture 1.3 - Krylov space methods - Arnoldi iteration

- **Krylov space:** $\text{span}\left(\hat{A}^0|\psi_0\rangle, \hat{A}^1|\psi_0\rangle, \hat{A}^2|\psi_0\rangle, \ldots, \hat{A}^{m-1}|\psi_0\rangle\right)$

$$|\psi_i\rangle \equiv \hat{A}|\psi_{i-1}\rangle \qquad \textit{ith - iteration}$$

$$h_{1,i} \equiv \langle\psi_0|\psi_i\rangle \quad |\psi_i\rangle \equiv |\psi_i\rangle - h_{1,i}|\psi_0\rangle$$

$$h_{2,i} \equiv \langle\psi_1|\psi_i\rangle \quad |\psi_i\rangle \equiv |\psi_i\rangle - h_{2,i}|\psi_1\rangle$$

$$\ldots$$

$$h_{i,i} \equiv \langle\psi_{i-1}|\psi_i\rangle \quad |\psi_i\rangle \equiv |\psi_i\rangle - h_{i+1,i}|\psi_{i-1}\rangle$$

$$h_{i+1,1} \equiv \| |\psi_i\rangle \| \quad |\psi_i\rangle \equiv |\psi_i\rangle / h_{i+1,1}$$

after i iterations, we have i+1 orthogonal vectors, i.e. we have the "semi-unitary" matrix

$$\mathbf{Q}_i = \begin{pmatrix} |\psi_0\rangle & |\psi_1\rangle & \ldots & |\psi_i\rangle \end{pmatrix}$$

$$D \times (i+1) \qquad \mathbf{Q}_i^\dagger \mathbf{Q}_i = \mathbb{1}$$

- What is the matrix **h?**

The iteration can be written in the matrix form:

$$\hat{A}\mathbf{Q}_i = \mathbf{Q}_{i+1}\mathbf{h}_i$$

- Terminating means:

$$\mathbf{Q}_{i+1} = \begin{pmatrix} |\psi_0\rangle & |\psi_1\rangle & \ldots & |\psi_i\rangle & \cancel{|\psi_{i+1}\rangle} \end{pmatrix}$$

$$\mathbf{Q}_{i+1}\mathbf{h}_i \approx \mathbf{Q}_i\tilde{\mathbf{h}}_i$$

$$\tilde{\mathbf{h}}_i = \begin{pmatrix} h_{1,1} & h_{1,2} & \ldots & h_{1,i} \\ h_{2,1} & h_{2,2} & \ldots & h_{2,i} \\ 0 & h_{3,2} & \ldots & h_{3,i} \\ \vdots & \vdots & \ddots & \vdots \\ \cancel{0} & \cancel{0} & \ldots & h_{i+1,i} \end{pmatrix}$$

- Then: $\hat{A}\mathbf{Q}_i \approx \mathbf{Q}_i\tilde{\mathbf{h}}_i$ $\qquad \tilde{\mathbf{h}}_i \approx \mathbf{Q}_i^\dagger \hat{A}\mathbf{Q}_i$ $\qquad \hat{A} \approx \mathbf{Q}_i\tilde{\mathbf{h}}_i\mathbf{Q}_i^\dagger$

- The matrix **h** is an approximation of the matrix **A** in (the much smaller) Krylov space!

Lecture 1.3 - Krylov space methods - Arnoldi iteration

- **Krylov space:** $\text{span}\left(\hat{A}^0\left|\psi_0\right\rangle, \hat{A}^1\left|\psi_0\right\rangle, \hat{A}^2\left|\psi_0\right\rangle, \ldots, \hat{A}^{m-1}\left|\psi_0\right\rangle\right)$

$$\left|\psi_i\right\rangle \equiv \hat{A}\left|\psi_{i-1}\right\rangle \qquad \textit{ith - iteration}$$

$$h_{1,i} \equiv \langle\psi_0|\psi_i\rangle \quad \left|\psi_i\right\rangle \equiv \left|\psi_i\right\rangle - h_{1,i}\left|\psi_0\right\rangle$$

$$h_{2,i} \equiv \langle\psi_1|\psi_i\rangle \quad \left|\psi_i\right\rangle \equiv \left|\psi_i\right\rangle - h_{2,i}\left|\psi_1\right\rangle$$

$$\ldots$$

$$h_{i,i} \equiv \langle\psi_{i-1}|\psi_i\rangle \quad \left|\psi_i\right\rangle \equiv \left|\psi_i\right\rangle - h_{i+1,i}\left|\psi_{i-1}\right\rangle$$

$$h_{i+1,1} \equiv \| \left|\psi_i\right\rangle \| \quad \left|\psi_i\right\rangle \equiv \left|\psi_i\right\rangle / h_{i+1,1}$$

after i iterations, we have i+1 orthogonal vectors, i.e. we have the "semi-unitary" matrix

$$\mathbf{Q}_i = \begin{pmatrix} \left|\psi_0\right\rangle & \left|\psi_1\right\rangle & \ldots & \left|\psi_i\right\rangle \end{pmatrix}$$

$$D \times (i+1) \qquad \mathbf{Q}_i^\dagger \mathbf{Q}_i = \mathbb{1}$$

- The matrix **h** is an approximation of the matrix **A** in (the much smaller) Krylov space!

- The idea is to now do exact diagonalization on this matrix and then to transform it back to the full space

- **Remark:** For hermitian matrices, e.g. $\hat{A} = \hat{H}$ the matrix **h** is even only tri-diagonal. Then the iteration to obtain **h** is even simpler, it's called **Lanczos** iteration. It's very useful to compute a few eigenvalues and eigenstates of very large sparse Hamiltonians.

$$\mathbf{h}_i = \begin{pmatrix} h_{1,1} & h_{1,2} & 0 & 0 & \ldots \\ h_{2,1} & h_{2,2} & h_{2,3} & 0 & \ldots \\ 0 & h_{3,2} & h_{3,2} & h_{3,2} & \ldots \\ 0 & 0 & h_{4,3} & h_{4,3} & \ldots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

Lecture 1.3 - Matrix exponential on Krylov space

- **Goal:** compute $\mathrm{e}^{-\mathrm{i}\hat{H}t}\,|\psi_0\rangle = \mathrm{e}^{\hat{A}t}\,|\psi_0\rangle$

$$
\begin{aligned}
&|\psi_i\rangle \equiv \hat{A}\,|\psi_{i-1}\rangle \\
&\quad h_{1,i} \equiv \langle\psi_0|\psi_i\rangle \quad |\psi_i\rangle \equiv |\psi_i\rangle - h_{1,i}\,|\psi_0\rangle \\
&\quad h_{2,i} \equiv \langle\psi_1|\psi_i\rangle \quad |\psi_i\rangle \equiv |\psi_i\rangle - h_{2,i}\,|\psi_1\rangle \\
&\qquad\qquad \cdots \\
&\quad h_{i,i} \equiv \langle\psi_{i-1}|\psi_i\rangle \quad |\psi_i\rangle \equiv |\psi_i\rangle - h_{i+1,i}\,|\psi_{i-1}\rangle \\
&h_{i+1,1} \equiv \|\,|\psi_i\rangle\,\| \quad |\psi_i\rangle \equiv |\psi_i\rangle\,/h_{i+1,1}
\end{aligned}
$$

$$
\mathbf{Q}_i = \begin{pmatrix} |\psi_0\rangle & |\psi_1\rangle & \cdots & |\psi_i\rangle \end{pmatrix}
$$

- For the matrix exponential applied to $|\psi_0\rangle$

… approximation is formally

$$
\mathrm{e}^{\hat{A}}\,|\psi_0\rangle \approx \mathbf{Q}_m \mathrm{e}^{\mathbf{h}} \mathbf{Q}_m^{\dagger}\,|\psi_0\rangle
$$

… but since $|\psi_0\rangle$ is the first row in $\mathbf{Q}_m^{\dagger}$, and all other rows are orthogonal, we just need to take the first column of $\mathrm{e}^{\mathbf{h}}$

```
# Compute approximation to exp(A)*psi0
# ... with m Krylov basis states
function arnoldi_exp(A, psi0, m)

    T = typeof(A[1]) # real or complex
    D = length(psi0)

    Q = zeros(T, D, m) # the projection matrix
    h = zeros(T, m, m) # Krylov projection of A

    Q[:,1] = psi0   # assumed normalized
    for ii = 1:(m-1)
        psi_i = A*Q[:,ii]
        for jj = 1:ii
            h[jj,ii] = Q[:,jj]' * psi_i
            psi_i -= h[jj,ii] .* Q[:,jj]
        end
        h[ii+1, ii] = norm(psi_i)
        Q[:,ii+1] = psi_i ./ h[ii+1, ii]
    end

    # now return the matrix exponential
    return Q * exp(h)[:,1]

end
```

Lecture 1.3 - Matrix exponential on Krylov space

- Test with full random Hamiltonian $\hat{H}$

$$\hat{A} = -\mathrm{i}\Delta t \hat{H}$$

```
D = 1000; m = 20
H = rand(ComplexF64, D,D); H += H'
A = -1im .* 0.1 .* H

psi0 = rand(ComplexF64, D); psi0 ./= norm(psi0)

@time psi_ed = exp(A)*psi0
@time psi_krylov = arnoldi_exp(A, psi0, m)
@show abs(psi_ed' * psi_krylov)^2
```

```
  0.676812 seconds (16 allocations: 91.584 MiB)
  0.007632 seconds (831 allocations: 12.931 MiB)
abs(psi_ed' * psi_krylov) ^ 2 = 0.9999999999999936
```

Krylov approximation is already exact up to machine precision with m ~ 20. Speed-up of two orders of magnitude. For sparse matrices even more!

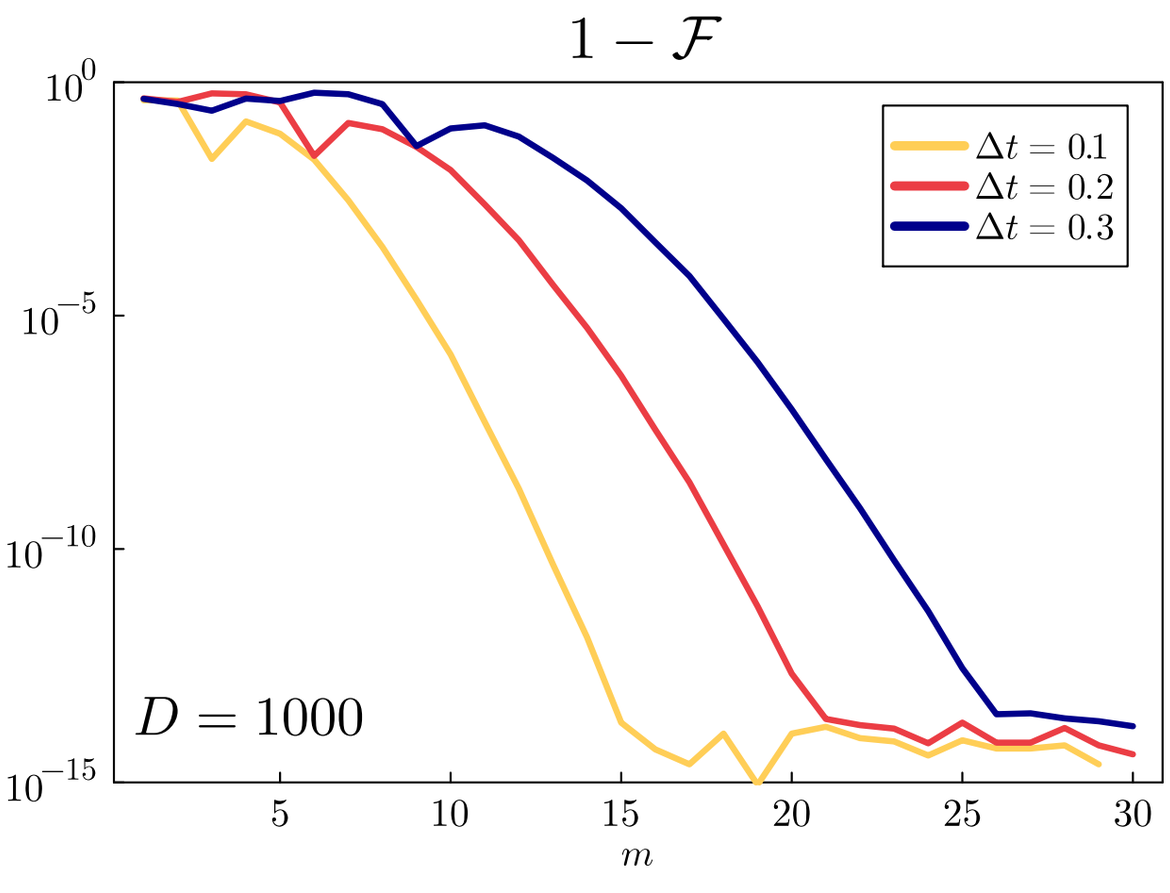
```
# Compute approximation to exp(A)*psi0
# ... with m Krylov basis states
function arnoldi_exp(A, psi0, m)

    T = typeof(A[1]) # real or complex
    D = length(psi0)

    Q = zeros(T, D, m) # the projection matrix
    h = zeros(T, m, m) # Krylov projection of A

    Q[:,1] = psi0   # assumed normalized
    for ii = 1:(m-1)
        psi_i = A*Q[:,ii]
        for jj = 1:ii
            h[jj,ii] = Q[:,jj]' * psi_i
            psi_i -= h[jj,ii] .* Q[:,jj]
        end
        h[ii+1, ii] = norm(psi_i)
        Q[:,ii+1] = psi_i ./ h[ii+1, ii]
    end

    # now return the matrix exponential
    return Q * exp(h)[:,1]

end
```

Lecture 1.3 - Matrix exponential on Krylov space

- Test with full random Hamiltonian $\hat{H}$

$$\hat{A} = -\mathrm{i}\Delta t \hat{H}$$

$$\mathcal{F} = |\langle\psi_0| (\mathrm{e}^{\hat{A}})^\dagger \mathbf{Q}_m \mathrm{e}^{\mathbf{h}} \mathbf{Q}_m^\dagger |\psi_0\rangle|^2$$



A larger "time-step" will need a larger Krylov space dimension.

```
# Compute approximation to exp(A)*psi0
# ... with m Krylov basis states
function arnoldi_exp(A, psi0, m)

    T = typeof(A[1]) # real or complex
    D = length(psi0)

    Q = zeros(T, D, m) # the projection matrix
    h = zeros(T, m, m) # Krylov projection of A

    Q[:,1] = psi0   # assumed normalized
    for ii = 1:(m-1)
        psi_i = A*Q[:,ii]
        for jj = 1:ii
            h[jj,ii] = Q[:,jj]' * psi_i
            psi_i -= h[jj,ii] .* Q[:,jj]
        end
        h[ii+1, ii] = norm(psi_i)
        Q[:,ii+1] = psi_i ./ h[ii+1, ii]
    end

    # now return the matrix exponential
    return Q * exp(h)[:,1]

end
```

Lecture 1.3 - Matrix exponential on Krylov space

- **Remark:** The implementation on the right is very simple, we did not implement any stopping condition and didn't discuss any (smart) error estimation

- There are many implementations out there, e.g. *KrylovKit* for Julia (or *ExpoKit*), etc.

```julia
using KrylovKit

psi, info = exponentiate(A, 1.0, psi0;
                         krylovdim=m, tol=1e-12)
```

```julia
# Compute approximation to exp(A)*psi0
# ... with m Krylov basis states
function arnoldi_exp(A, psi0, m)

    T = typeof(A[1]) # real or complex
    D = length(psi0)

    Q = zeros(T, D, m) # the projection matrix
    h = zeros(T, m, m) # Krylov projection of A

    Q[:,1] = psi0   # assumed normalized
    for ii = 1:(m-1)
        psi_i = A*Q[:,ii]
        for jj = 1:ii
            h[jj,ii] = Q[:,jj]' * psi_i
            psi_i -= h[jj,ii] .* Q[:,jj]
        end
        h[ii+1, ii] = norm(psi_i)
        Q[:,ii+1] = psi_i ./ h[ii+1, ii]
    end

    # now return the matrix exponential
    return Q * exp(h)[:,1]

end
```

Lecture 1.3 - Applying Krylov space evolution in practice

- Long-range transverse Ising model: $\hat{H}_{\mathrm{TI}} = \sum_{i<j} J_{i,j} \hat{\sigma}_i^z \hat{\sigma}_j^z + h_x \sum_i \hat{\sigma}_i^x \qquad J_{ij} = \frac{J}{|i-j|^\alpha}$

```
Jij = zeros(N,N)
alpha = 3
for ii = 1:N
    for jj = (ii+1):N
        Jij[ii,jj] = 1/(abs(ii-jj)^alpha)
    end
end
```

… sparse, as discussed earlier

```
H = spzeros(2^N, 2^N)
for ii = 1:N
    H += hx .* (sms[ii]' + sms[ii]) # sx
    szii = (sms[ii]'*sms[ii] - sms[ii]*sms[ii]') # sz spin ii
    for jj = (ii+1):N
        szjj = (sms[jj]'*sms[jj] - sms[jj]*sms[jj]') # sz spin jj
        H += Jij[ii,jj] .* szii * szjj
    end
end
```

- Initial state (highly excited, not an eigenstate)

Néel state

$$|\psi_0\rangle = |\downarrow\uparrow\downarrow\uparrow\downarrow\uparrow \ldots\rangle$$

```
psi = zeros(2^N)
psi[1] = 1.0
for oo = 1:2:N
    psi = sms[oo]' * psi
end
```

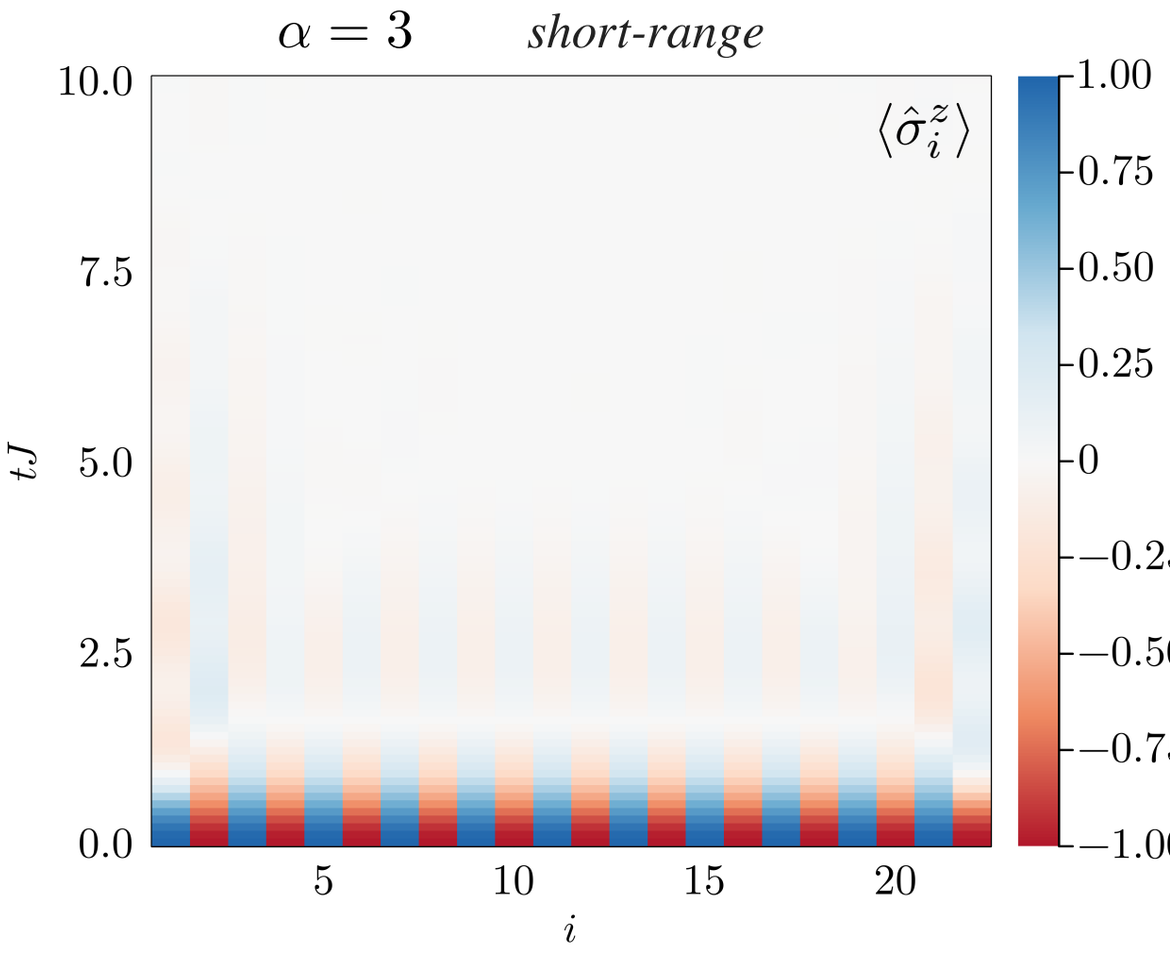
- Then, applying Krylov time-evolution steps

```
psi, info = exponentiate(H, -1im * dt, psi)
```

Lecture 1.3 - Spin-model evolution with Krylov space matrix exponential

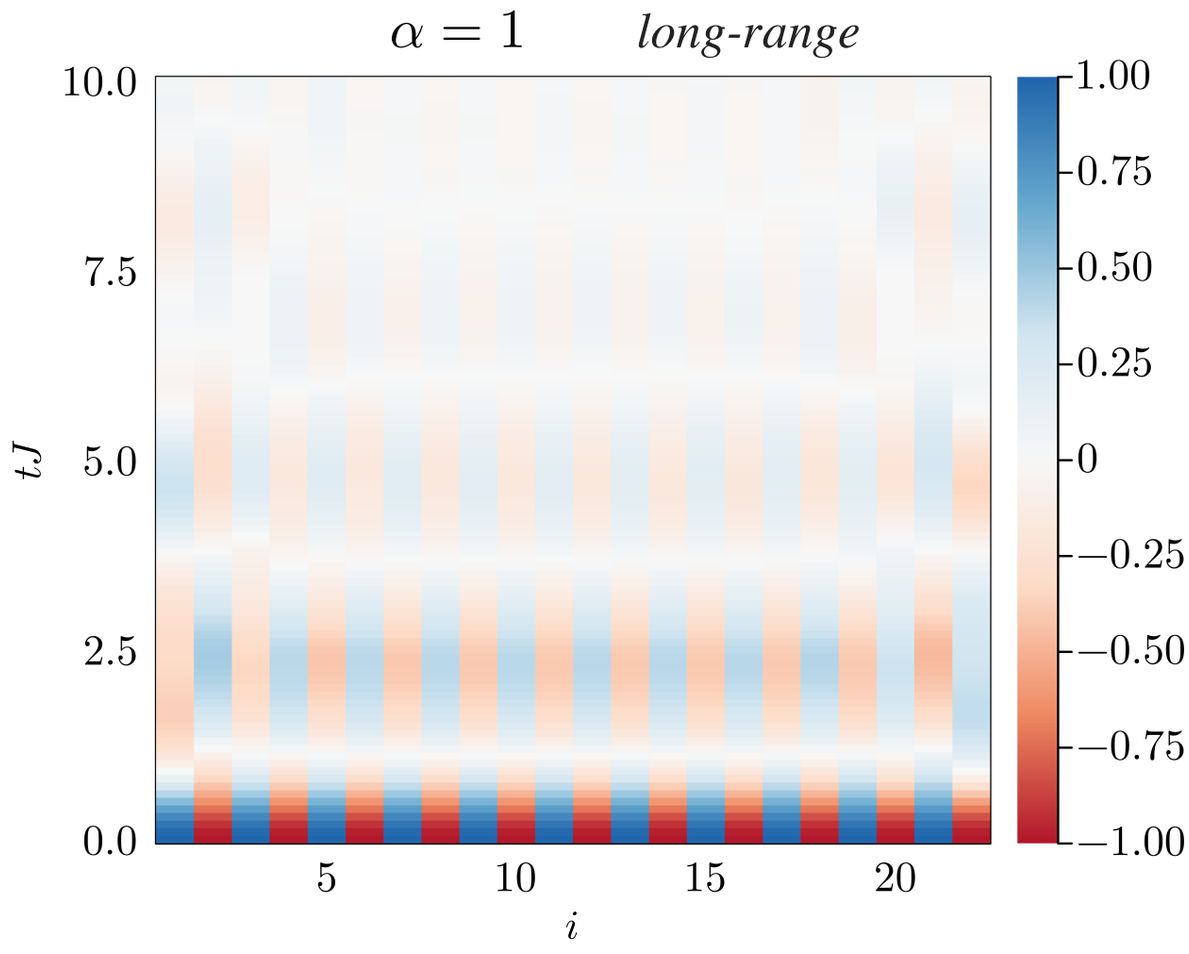
- Evolution of local spin-z component
- $N = 22$ spins

$$\hat{H}_{\mathrm{TI}} = \sum_{i<j} J_{i,j}\hat{\sigma}_i^z\hat{\sigma}_j^z + h_x\sum_i \hat{\sigma}_i^x \qquad J_{ij} = \frac{J}{|i-j|^\alpha} \qquad h_x = J$$



For the short range case, correlation build-up leads to local mixed states. For example, for sufficiently long times, the reduced density matrices will become fully mixed.

$$\hat{\rho}_i = \mathrm{tr}_{j\neq i}(|\psi(t)\rangle\langle\psi(t)|) \approx \begin{pmatrix} 1/2 & 0 \\ 0 & 1/2 \end{pmatrix} \qquad \mathrm{tr}(\hat{\rho}_i\hat{\sigma}_i^z) \approx 0 \qquad t \gg J^{-1}$$



In the long-range case, dynamics is more collective

$$\alpha = 0$$

$$\hat{H}_{\mathrm{TI}} = J\sum_{i<j} \hat{\sigma}_i^z\hat{\sigma}_j^z + h_x\sum_i \hat{\sigma}_i^x$$

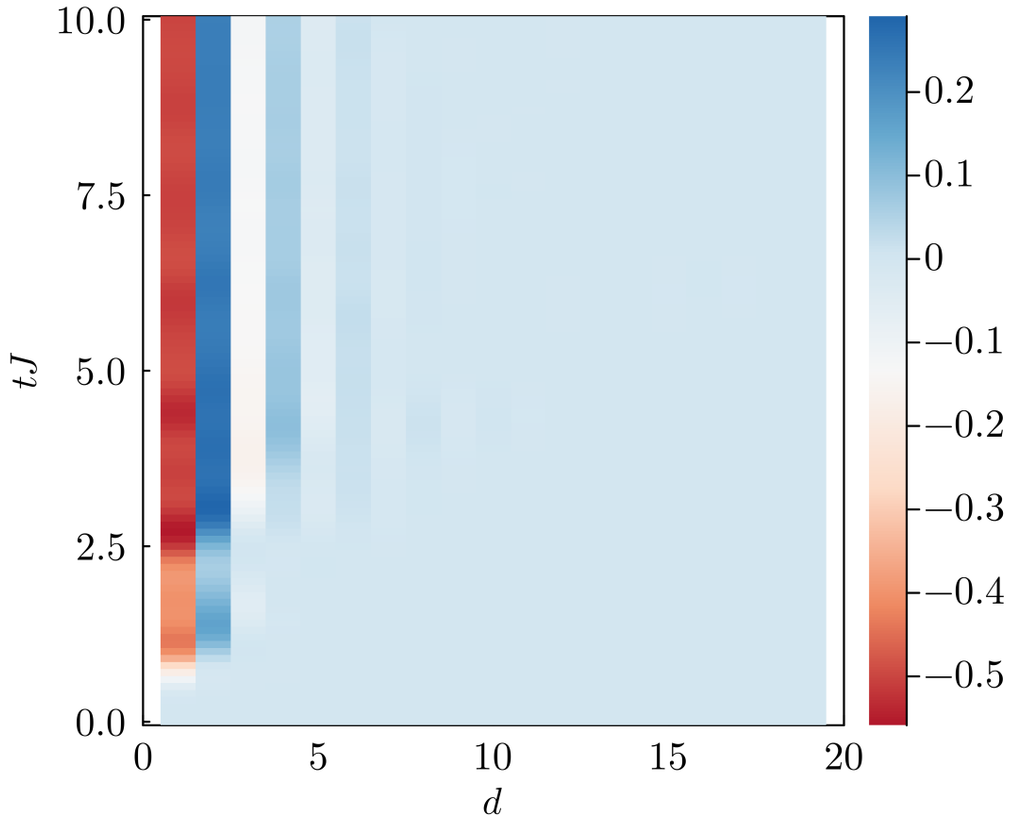
Lecture 1.3 - Spin-model evolution with Krylov space matrix exponential

- Evolution of local spin-z component
- $N = 22$ spins

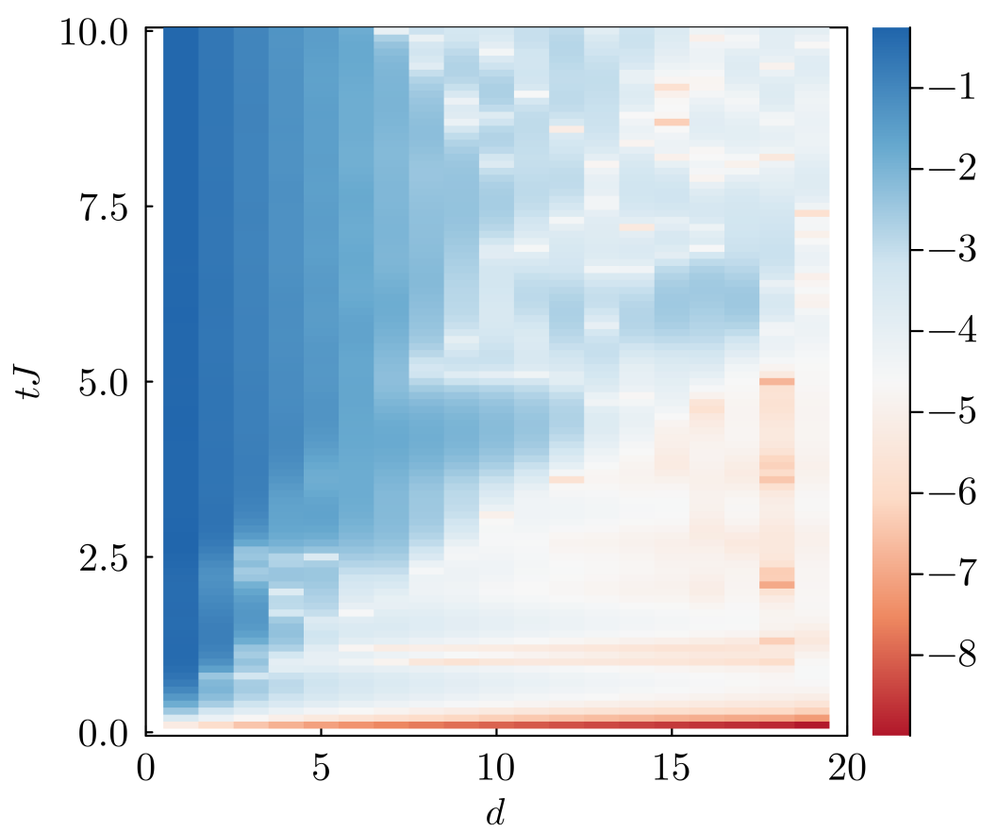
$$\hat{H}_{\mathrm{TI}} = \sum_{i<j} J_{i,j}\hat{\sigma}_i^z\hat{\sigma}_j^z + h_x\sum_i \hat{\sigma}_i^x \qquad J_{ij} = \frac{J}{|i-j|^\alpha}$$

$$C_{i,j} = \langle\hat{\sigma}_i^z\hat{\sigma}_j^z\rangle - \langle\hat{\sigma}_i^z\rangle\langle\hat{\sigma}_j^z\rangle$$

$\alpha = 3$ *short-range* $C_{3,3+d}$



$\log_{10}(|C_{3,3+d}|)$



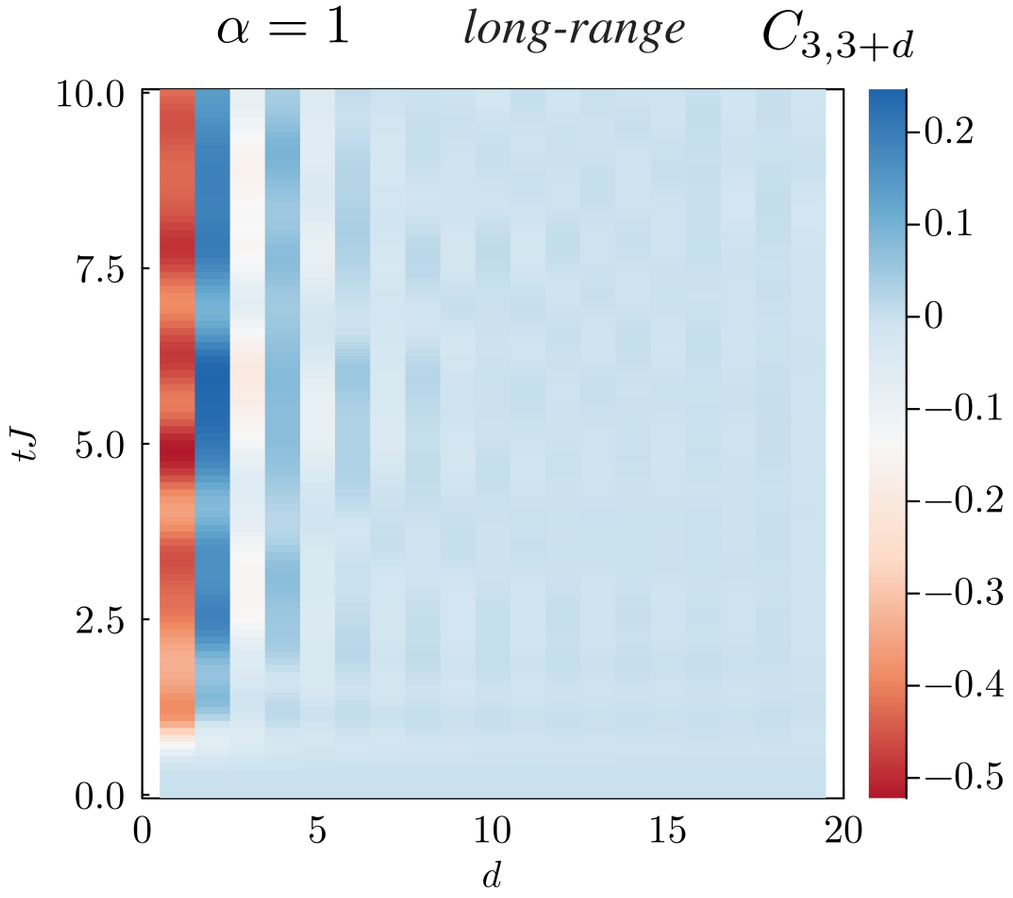
There is a spreading out in a "light-cone". Even the two-spin correlations become very small over long-distances.

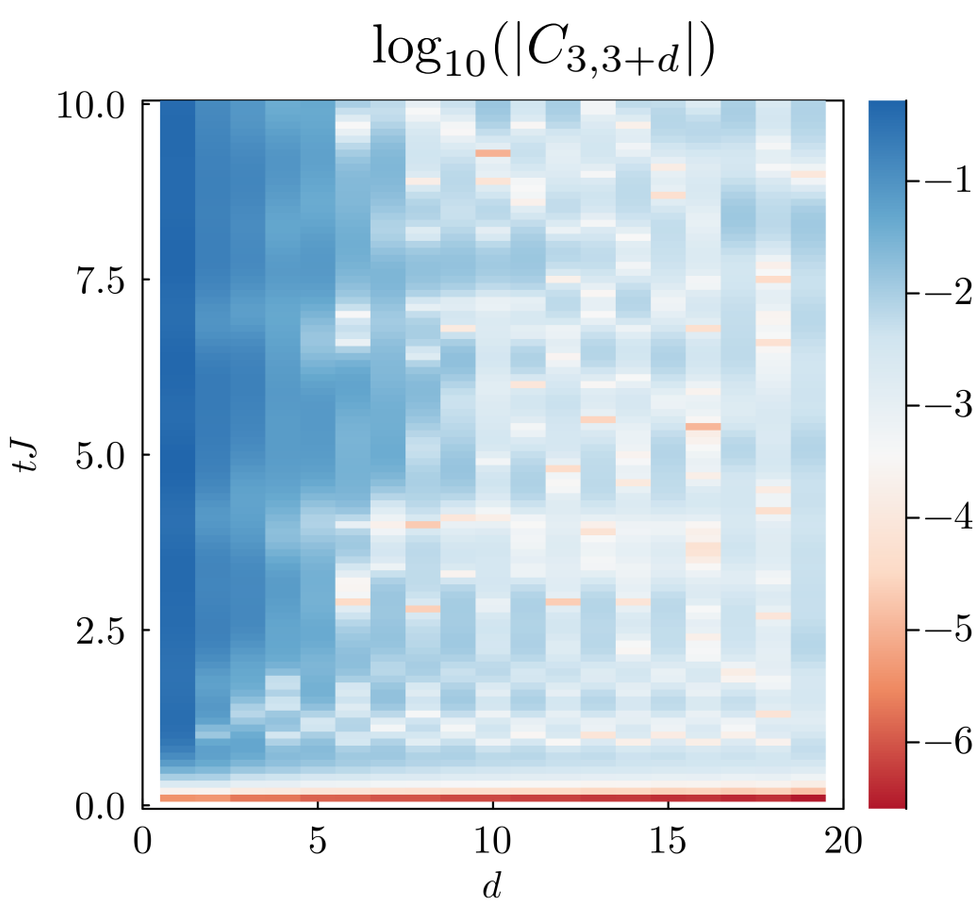
Lecture 1.3 - Spin-model evolution with Krylov space matrix exponential

- Evolution of two-spin correlations
- $N = 22$ spins

$$\hat{H}_{\mathrm{TI}} = \sum_{i<j} J_{i,j}\hat{\sigma}_i^z\hat{\sigma}_j^z + h_x \sum_i \hat{\sigma}_i^x \qquad J_{ij} = \frac{J}{|i-j|^\alpha}$$

$$C_{i,j} = \langle\hat{\sigma}_i^z\hat{\sigma}_j^z\rangle - \langle\hat{\sigma}_i^z\rangle\langle\hat{\sigma}_j^z\rangle$$





For long-range couplings the light-cone disappears

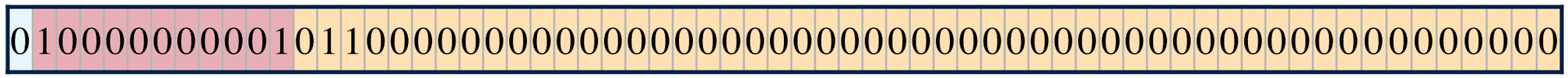
- Complex information spreading physics as competition between long and short-range terms … further reading:

P. Calabrese and J. Cardy, J. Stat. Mech, P04010 (2005)
J. Eisert, M. van den Worm, S. R. Manmana, M. Kastner, Phys. Rev. Lett. 111, 260401 (2013)
J. Schachenmayer, B. P. Lanyon, C. F. Roos, A. J. Daley, Phys. Rev. X 3, 031015 (2013)
Z.-X. Gong, M. Foss-Feig, Fernando G. S. L. Brandão, and A. V. Gorshkov, Phys. Rev. Lett. 119 050501 (2017)
L. Cevolani, J. Despres, G. Carleo, L. Tagliacozzo, L. Sanchez-Palencia, Phys. Rev. B 98, 024302 (2018)
I. Frérot, P. Naldesi, and T. Roscilde, Phys. Rev. Lett. 120, 050401 (2018), ….

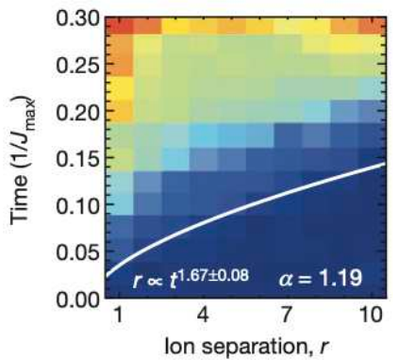
Lecture 1 - Plan for today

Today mostly: Tutorial style

- **Part 1.1:** Some fundamentals about numbers in digital memory and the linear algebra of quantum mechanics



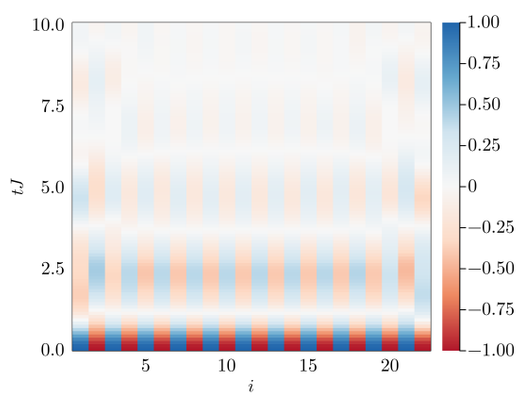
- **Part 1.2:** Constructing (sparse) Hamiltonians for many-body spin models



$$\cdots \otimes \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \otimes \cdots$$

spin #i *spin #j*

- **Part 1.3:** Krylov space methods. Applications to spin-model dynamics

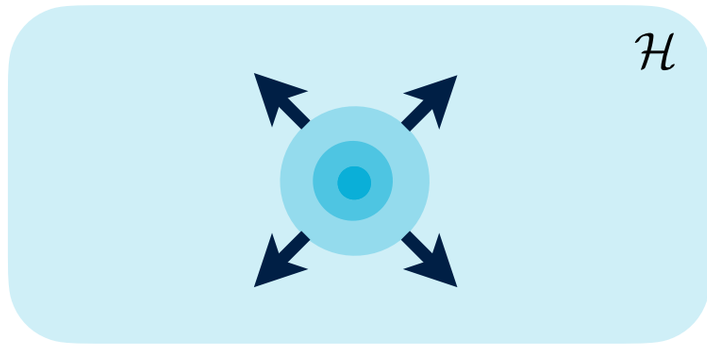


- **Part 1.4:** Quick discussion: How to go to larger systems?

Lecture 1.4 - How to simulate larger systems?

- How to beat the exponential Hilbert size growth?

  Mostly not all information of the state needed!



- **Compress the state** = Use a smaller sized state to approximate the full state:

- **Product state (mean-field)**
  (no entanglement)

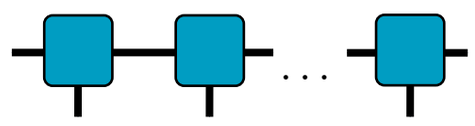
$$|\psi\rangle = |\phi_1\rangle\,|\phi_2\rangle \ldots |\phi_N\rangle$$

$\propto N$ coupled non-linear equations
… instead of $\propto 2^N$ linear ones

Sometimes excellent approximation (especially for high connectivity) ***(see lecture 3)***

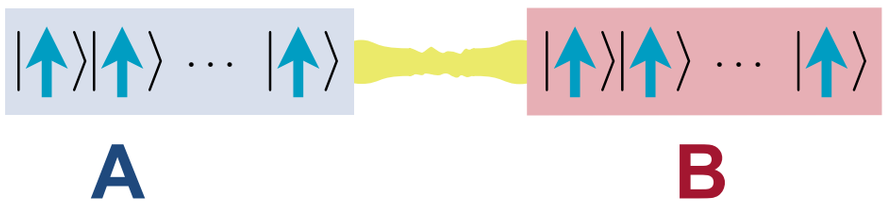
- **Matrix product state (MPS) … more generally: tensor networks**

$$|\psi\rangle \approx \mathrm{tr} \begin{pmatrix} |\psi_1^{11}\rangle & |\psi_1^{12}\rangle \\ |\psi_1^{21}\rangle & |\psi_1^{22}\rangle \\ & \ddots \end{pmatrix} \begin{pmatrix} |\psi_2^{11}\rangle & |\psi_2^{12}\rangle \\ |\psi_2^{21}\rangle & |\psi_2^{22}\rangle \\ & \ddots \end{pmatrix} \cdots \begin{pmatrix} |\psi_N^{11}\rangle & |\psi_N^{12}\rangle \\ |\psi_N^{21}\rangle & |\psi_N^{22}\rangle \\ & \ddots \end{pmatrix} \leftrightarrow$$



(lectures: Bañuls, Chepiga, Loïc, Pollmann, Waintal…)

General idea



$$|\psi\rangle = \sum_{i,j=1}^{D_A,D_B} c_{i,j}\,|i\rangle_A |j\rangle_B$$

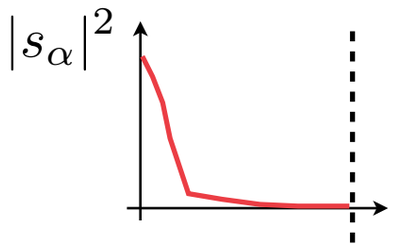
Compress this matrix!

Lecture 1.4 - How to simulate larger systems?

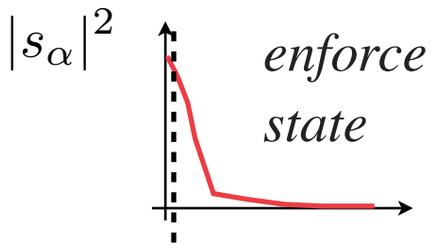
Compression with truncated singular value (Schmidt) decomposition

$$|\psi\rangle = \sum_{i,j=1}^{D_A,D_B} c_{i,j}|i\rangle_A|j\rangle_B \overset{\searrow}{=} \sum_{\alpha}^{\chi} s_\alpha |\alpha\rangle_A |\alpha\rangle_B \qquad (\mathbf{C})_{i,j} = c_{i,j} \qquad \mathbf{C} = \mathbf{USV}^\dagger$$

- Truncation at various levels *Matrix of gray-scale values in image*

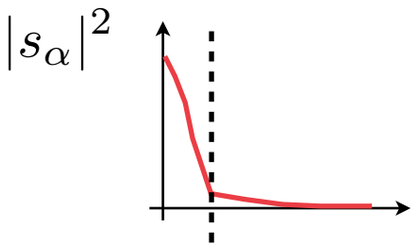


$\chi = D_A = D_B$



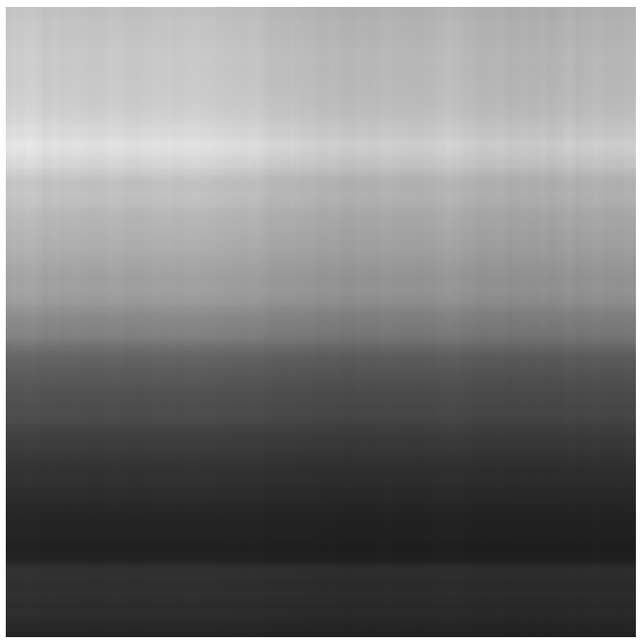
enforce product state

$\chi = 1$



$\chi = 0.03 D_A$

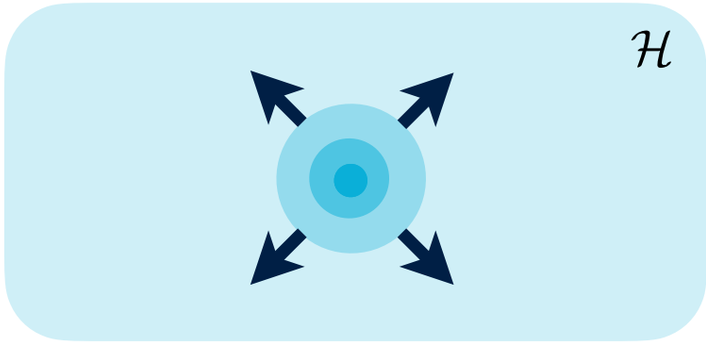
(not) Mont Blanc



mean-field (not) Mont Blanc

slightly entangled (not) Mont Blanc

Lecture 1.4 - How to simulate larger systems?



- How to beat the exponential Hilbert size growth?

Mostly not all information of the state needed!

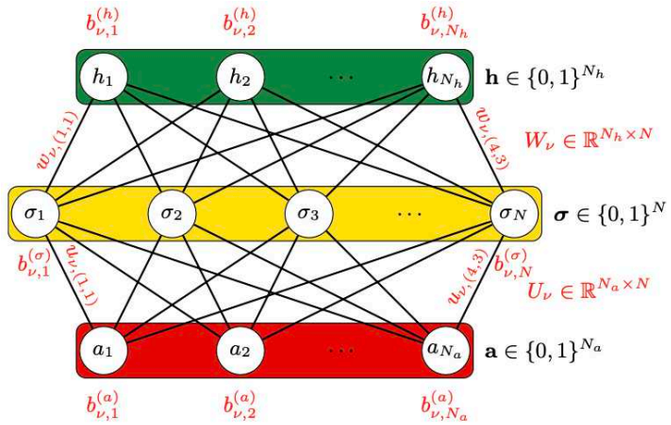
- **Use ansatz states in combination with variational methods**

Analytically inspired ansatz states (e.g. Jastrow, valence bond, entangled plaquettes, etc.)

$$\psi \sim S(\mathbf{r}_1 \cdots \mathbf{r}_N) \prod_{i<j=1}^{N} f(r_{ij})$$

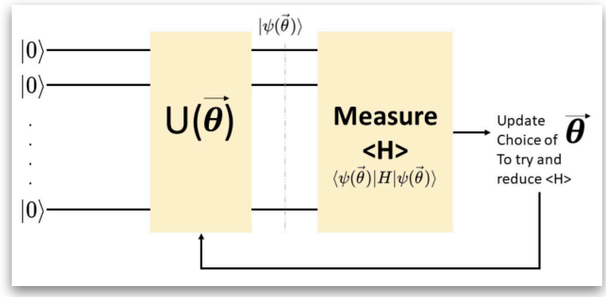
(lectures: Becca, …)

Neural network ansatz for quantum states



(lectures: Heyl, Vicentini, …)

Actual states on quantum computers:



(lectures: Ayral, …)

- **Do not simulate states, but only observables of interest:**

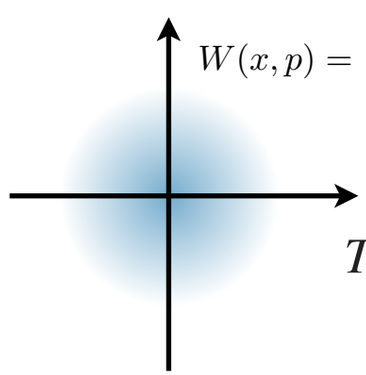
e.g. Path integral Monte-Carlo methods (drawing samples of wave-functions)

e.g. (Non-equilibrium) Green's function methods (diagrammatic solutions)

(lectures: Michel, Waintal, …)

Phase space!

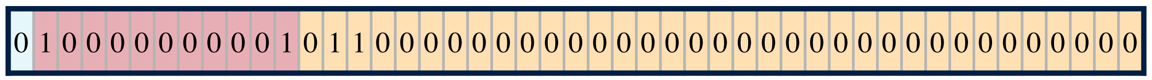
Rest of this lecture series

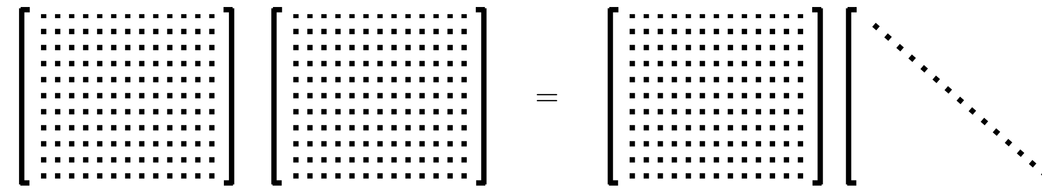


Truncated Wigner Approximation

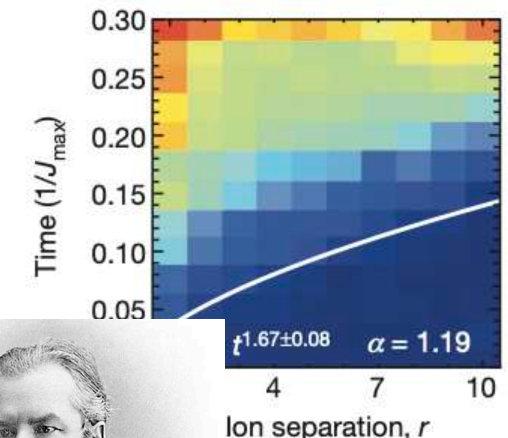
Lecture 1 - Recap

- We discussed how to represent numbers as bits. We describe systems with state-vectors (not only in QM). If the problem is linear (QM), an exact diagonalization of the Hamiltonian solves everything, and is easily doable numerically for small systems (of order 10 spins)





- We discussed some general spin-model physics with long-range couplings, as they can e.g. be engineered in trapped ion systems. We discussed how to construct Hamiltonians using Kronecker products. It's very important to use **sparse matrices**.

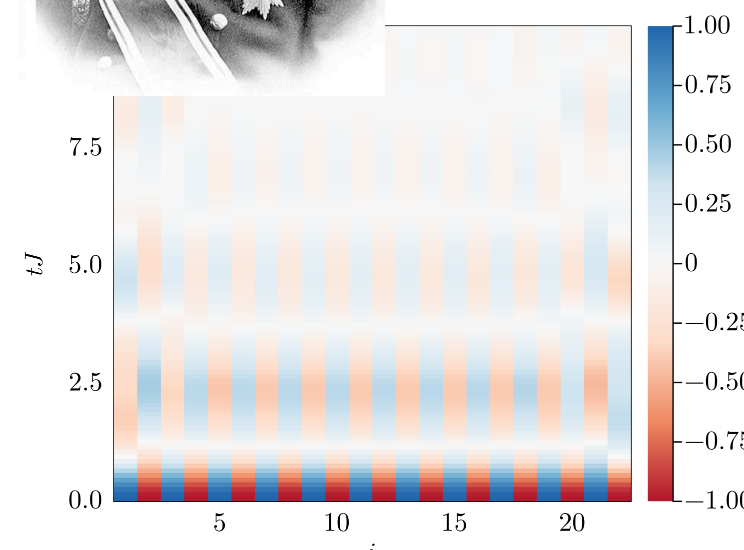


$$\hat{\sigma}_i^- = \begin{pmatrix} 1 & & \\ & 1 & \\ & & \ddots \end{pmatrix} \otimes \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \otimes \begin{pmatrix} 1 & & \\ & 1 & \\ & & \ddots \end{pmatrix}$$

- We introduced a time-evolution algorithm for linear systems, based on: **Krylov space**. Krylov space is a vector-space constructed from an initial state and the evolution matrix:

$$\text{span}\left(\hat{A}^0 \left|\psi_0\right\rangle, \hat{A}^1 \left|\psi_0\right\rangle, \hat{A}^2 \left|\psi_0\right\rangle, \ldots, \hat{A}^{m-1} \left|\psi_0\right\rangle\right)$$

Eigenvectors need to be made orthonormal, using **Arnoldi** or **Lanczos** iterations, then diagonalizations and matrix exponentials can be performed very efficiently on the much smaller **Krylov space.** This allows to easily simulate quantum dynamics of ~22 spins/qubits on a laptop.



- We briefly discussed how to push simulations to larger systems … next we go to **phase space!**