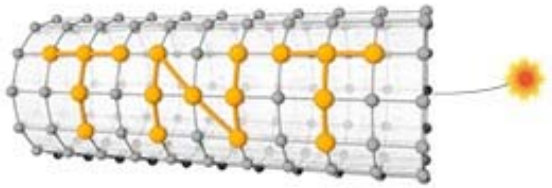


Tensor Network Theory

An introduction to DMRG and MPS methods

Quantum and Kinetic Problems: Modeling, Analysis, Numerics and Applications
IMS Singapore

Dieter Jaksch

University of Oxford

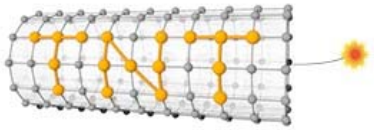
Special thanks to: Stephen Clark, University of Bristol

UNIVERSITY OF OXFORD

Q-MAC

CQT

EPSRC



Outline of lectures

- Lecture 1

Introduction to strong correlations, many-body problem, recap on essential linear algebra we will need later.

- Lecture 2

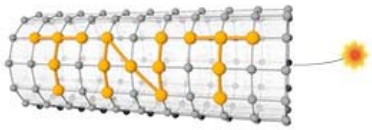
Tensors and contractions, product states and variational principle, matrix product states (MPS), and their entanglement properties.

- Lecture 3

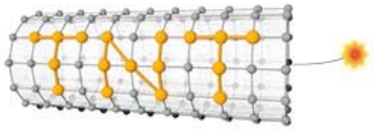
The “calculus” of MPS, algorithms for their variational optimisation, and algorithms for time-evolution of MPS.

- Lecture 4

Moving to finite temperatures, renormalisation approaches to tensor networks, extension to 2D with projected entangled pairs.



1: Calculus of MPS

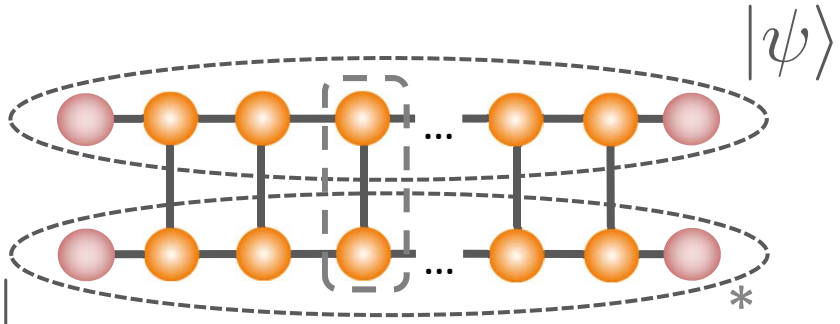


Calculus of MPS

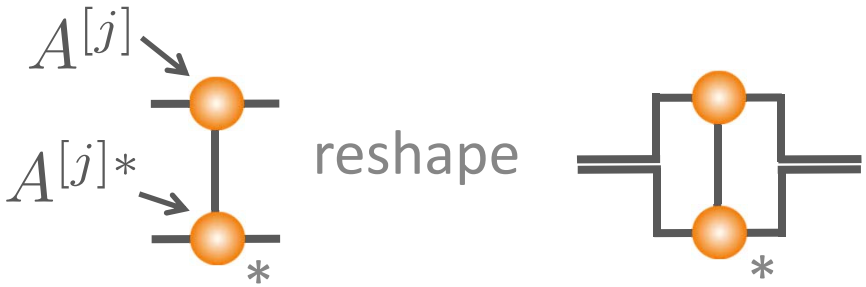
The key property of MPS tensor network is that many calculations become very efficient:

Norms and overlaps:

$$\langle\psi|\psi\rangle \;=$$



A useful object in this network is:



$$\mathbf{E}^{[j]} = \sum_{\sigma_j} \mathbf{A}^{[j]\sigma_j} \left(\mathbf{A}^{[j]\sigma_j}\right)^*$$

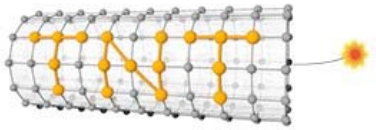
called the "transfer matrix"

The norm is then:

$$\langle\psi|\psi\rangle = \vec{E}^{[1]}\mathbf{E}^{[2]}\mathbf{E}^{[3]}\ldots\mathbf{E}^{[N-1]}\vec{E}^{[N]}$$

$$(1\times\chi^2)(\chi^2\times\chi^2)\quad(\chi^2\times\chi^2)(\chi^2\times 1)$$

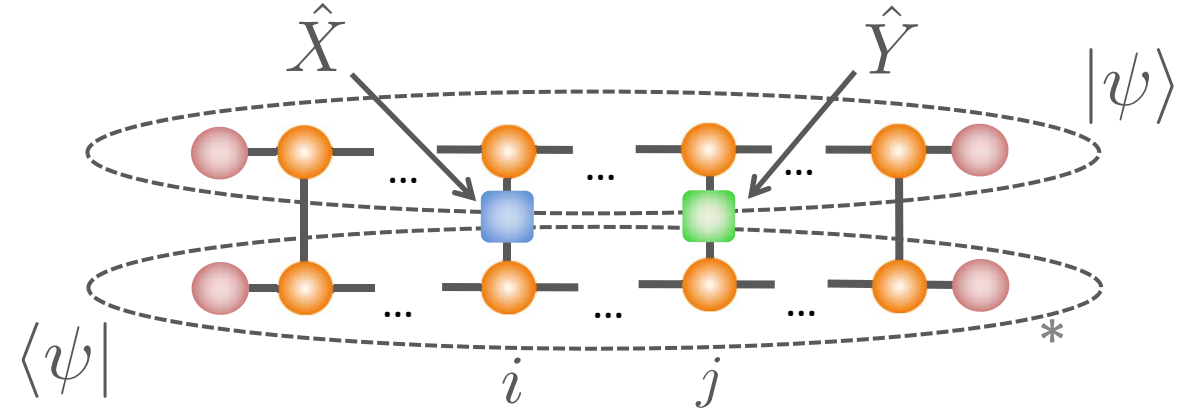
This is the multiplication of $(1\times\chi^2)$ with $N-2$ $(\chi^2\times\chi^2)$ matrices with a complexity that scales as: $\mathcal{O}(N\chi^4)$ → **efficient!**



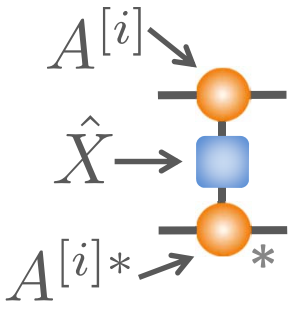
Calculus of MPS

Expectation values:

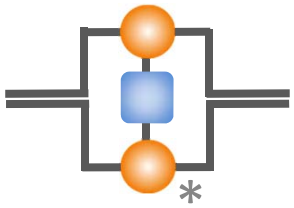
$$\langle\psi|\hat{X}_i\hat{Y}_j|\psi\rangle \;=$$



In addition to the transfer matrices we now also have:



reshape



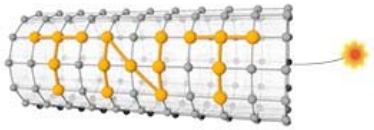
$$\mathbf{E}_X^{[i]} = \sum_{\sigma_i,\sigma_i'} \mathbf{A}^{[i]\sigma_i} X_{\sigma_i\sigma_i'} (\mathbf{A}^{[i]\sigma_i'})^*$$

Similarly for $\hat{Y}$

The expectation value is then a product of vectors and matrices:

$$\langle\psi|\hat{X}_i\hat{Y}_j|\psi\rangle = \vec{E}^{[1]}\mathbf{E}^{[2]}\ldots\mathbf{E}^{[i-1]}\mathbf{E}_X^{[i]}\mathbf{E}^{[i+1]}\ldots\mathbf{E}^{[j-1]}\mathbf{E}_Y^{[j]}\mathbf{E}^{[j+1]}\ldots\mathbf{E}^{[N-1]}\vec{E}^{[N]}$$

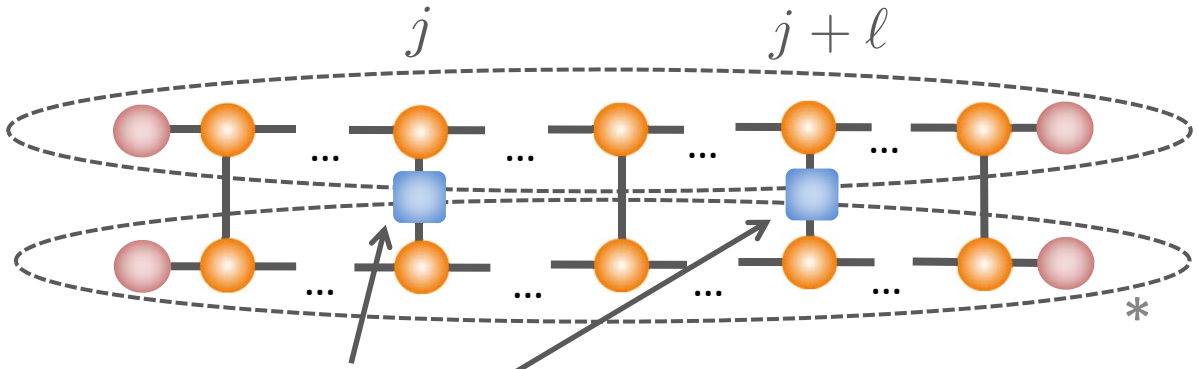
This has the same complexity as the norm $\mathcal{O}(N\chi^4)$ → **efficient!**



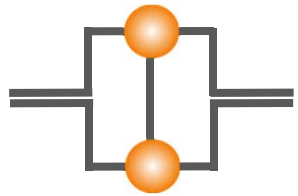
Calculus of MPS

Correlation lengths:

The transfer matrix, which appears many times, contains crucial information about how correlations behaviour in general in MPS:



Diagonalise



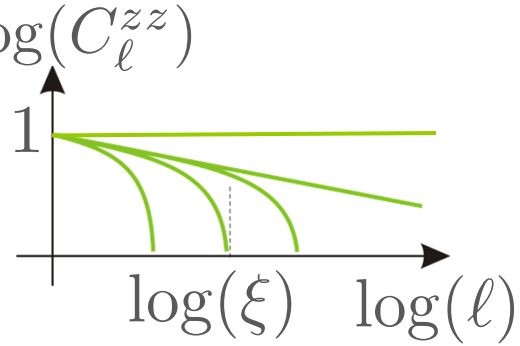
eigenvalues: ζ_μ

corr. lengths: $\xi_\mu = 1/\log(|\zeta_\mu|)$

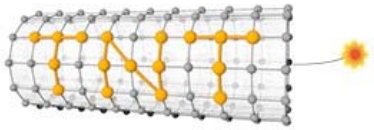
$$C_\ell^{zz} = \langle\psi|\sigma_j^z\sigma_{j+\ell}^z|\psi\rangle - \langle\psi|\sigma_j^z|\psi\rangle\langle\psi|\sigma_{j+\ell}^z|\psi\rangle$$

One can show that :

$$C_\ell^{zz} = \sum_{\mu=2}^{\chi^2} \kappa_\mu \left(\frac{\zeta_\mu}{|\zeta_\mu|}\right)^\ell e^{-\ell/\xi_\mu}$$



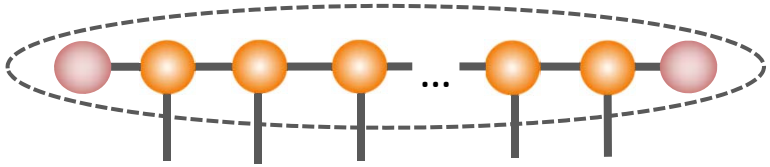
Once χ is fixed and finite MPS always have exponentially decaying correlations, but can still model algebraic on short length scales.



Gauge freedom

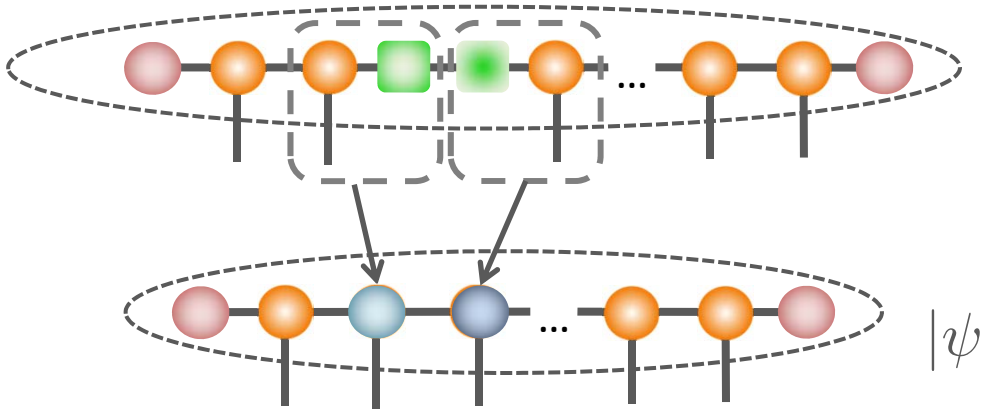
An MPS representation of a state is not-unique. Given any invertible square matrix $\mathbf{X}$ we can simply insert it and its inverse on any internal leg without changing the state:

$$|\psi\rangle = \sum_{\vec{\sigma}} \mathbf{A}^{\sigma_1}\mathbf{A}^{\sigma_2}\mathbf{A}^{\sigma_3}\cdots\mathbf{A}^{\sigma_N}|\vec{\sigma}\rangle$$



$$|\psi\rangle = \sum_{\vec{\sigma}} \mathbf{A}^{\sigma_1}\mathbf{A}^{\sigma_2}\mathbf{X}\mathbf{X}^{-1}\mathbf{A}^{\sigma_3}\cdots\mathbf{A}^{\sigma_N}|\vec{\sigma}\rangle$$

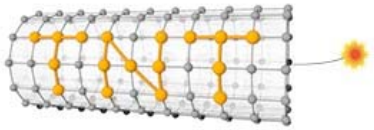
$$\mathbf{X}\mathbf{X}^{-1} = \mathbb{1}$$



We obtain a different MPS of the same dimension:

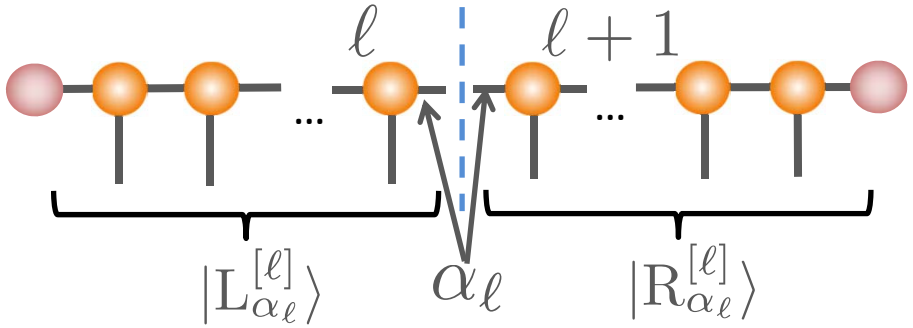
$$|\psi\rangle = \sum_{\vec{\sigma}} \mathbf{A}^{\sigma_1}\tilde{\mathbf{A}}^{\sigma_2}\tilde{\mathbf{A}}^{\sigma_3}\cdots\mathbf{A}^{\sigma_N}|\vec{\sigma}\rangle$$

This “gauge freedom” can be exploited to establish a crucial property for stable algorithms – *orthogonality …*



Orthogonality

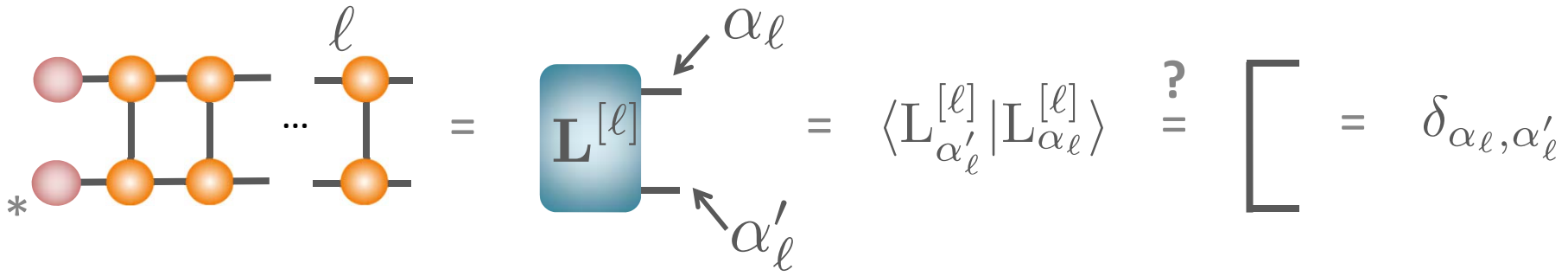
Let's take an MPS and split some internal leg into two pieces …



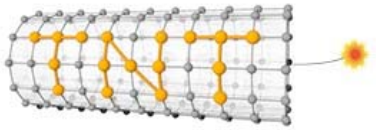
is equivalent to the form:

$$|\psi\rangle = \sum_{\alpha_\ell} |\mathrm{L}^{[\ell]}_{\alpha_\ell}\rangle |\mathrm{R}^{[\ell]}_{\alpha_\ell}\rangle$$

What are the properties of the states $|\mathrm{L}^{[\ell]}_{\alpha_\ell}\rangle$ and $|\mathrm{R}^{[\ell]}_{\alpha_\ell}\rangle$ of the left and right subsystems? In particular are they orthogonal? Consider:

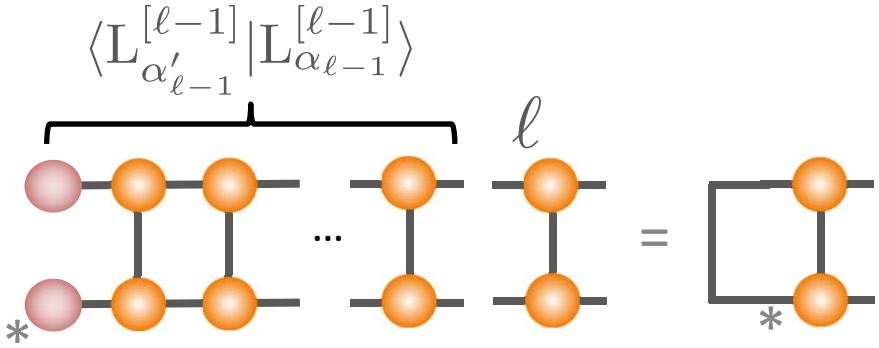


which is an overlap matrix for the set of "left" states. If this is the identity matrix then they are an orthonormal basis of a subspace.



Orthogonality

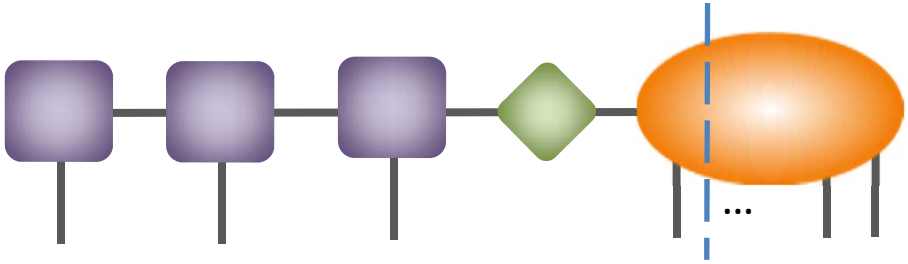
Suppose that the "left" states at the splitting $\ell - 1$ are orthogonal, then



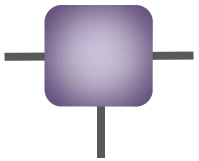
so for $|\mathrm{L}^{[\ell]}_{\alpha_\ell}\rangle$ to be orthogonal we need:

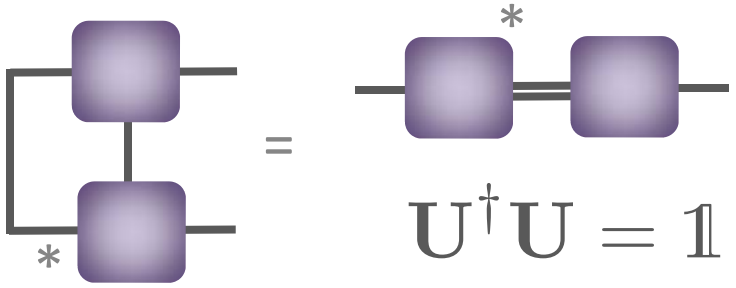
$$\sum_{\sigma_\ell} (\mathbf{A}^{[\ell]\sigma_\ell})^\dagger \mathbf{A}^{[\ell]\sigma_\ell} = \mathbb{1}$$

Thus we are left with a local condition on the $\mathbf{A}^{[\ell]}$ matrices. If this is obeyed by all the matrices $\mathbf{A}^{[k]}$ for sites $k \leq \ell$ then all their left states are orthogonal. This is exactly what we want for an *exact* MPS:

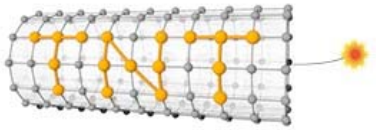


they came from a reduced SVD, so they automatically obey:

each  is "unitary"



$$\mathbf{U}^\dagger \mathbf{U} = \mathbb{1}$$

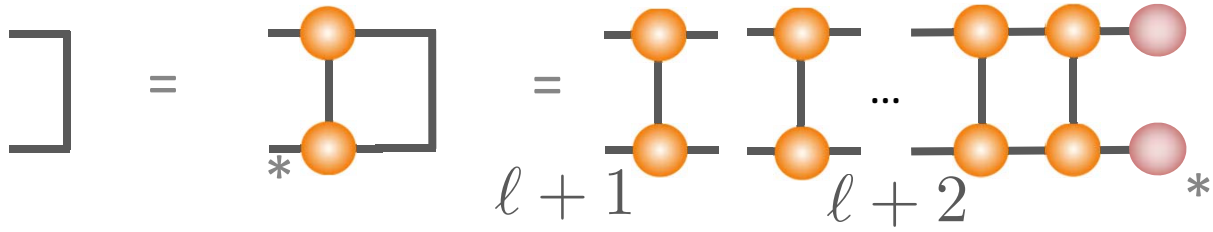


Orthogonality

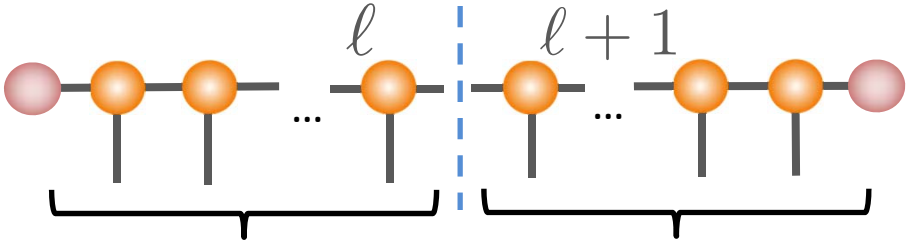
A similar condition applies to the "right" states of our splitting:

$$\sum_{\sigma_{\ell+1}} \mathbf{A}^{[\ell+1]\sigma_{\ell+1}} \left(\mathbf{A}^{[\ell+1]\sigma_{\ell+1}}\right)^\dagger = \mathbb{1}$$

We say this is *right* orthonormalised

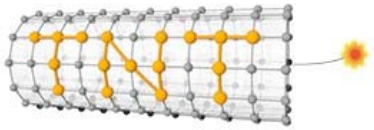


If this is obeyed by all the matrices $\mathbf{A}^{[k]}$ for sites $k > \ell$ then all their right states are orthogonal. So how do we turn **any** MPS into a form with left-right orthogonality about some split?



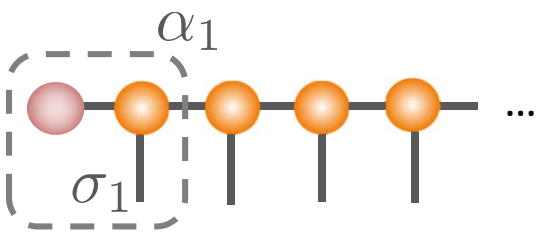
all *left* orthonormalised all *right* orthonormalised

Apply same strategy as for the exact MPS construction …



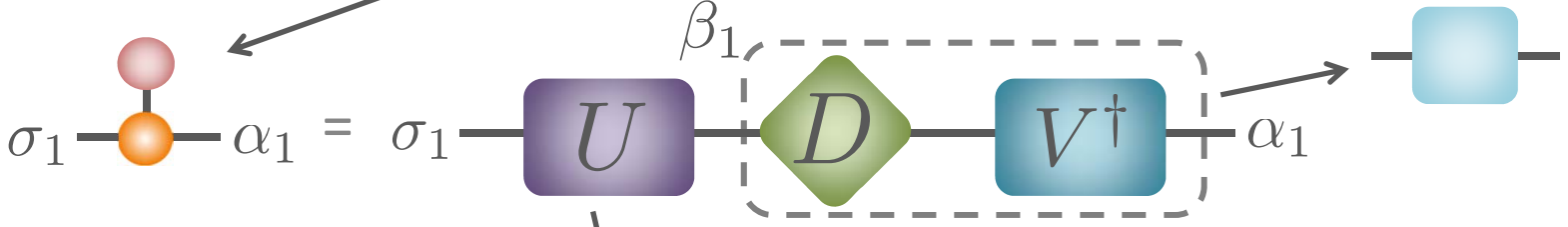
Enforcing orthogonality

Start from the left:

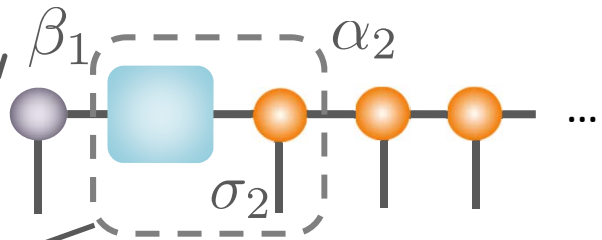


SVD …

collect up remainder:

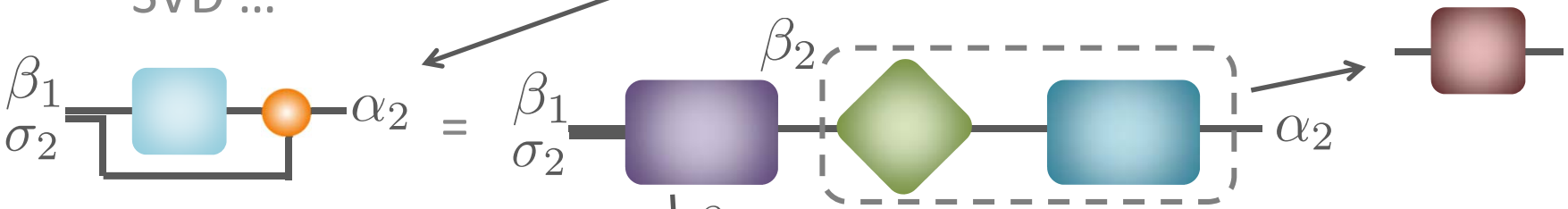


new **A** matrix is left orthonormalised:

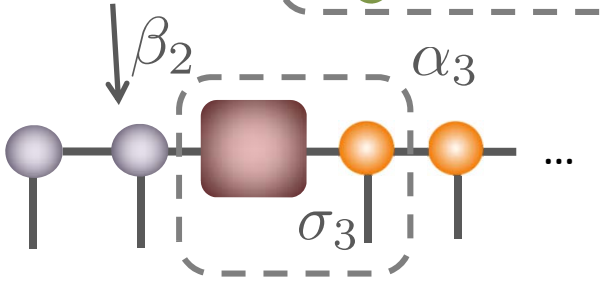


move to next site:

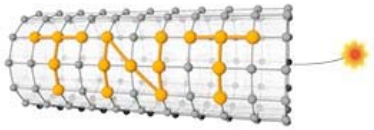
SVD …



next **A** matrix is left orthonormalised:

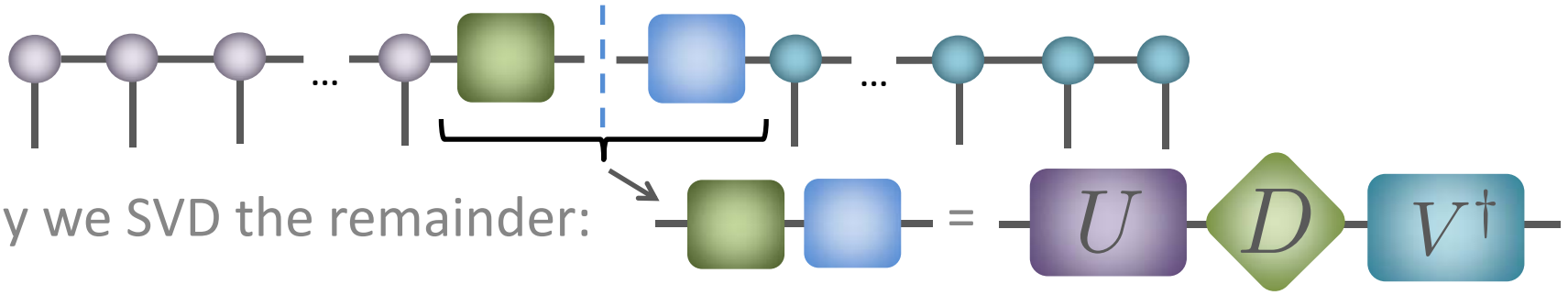


repeat:



Enforcing orthogonality

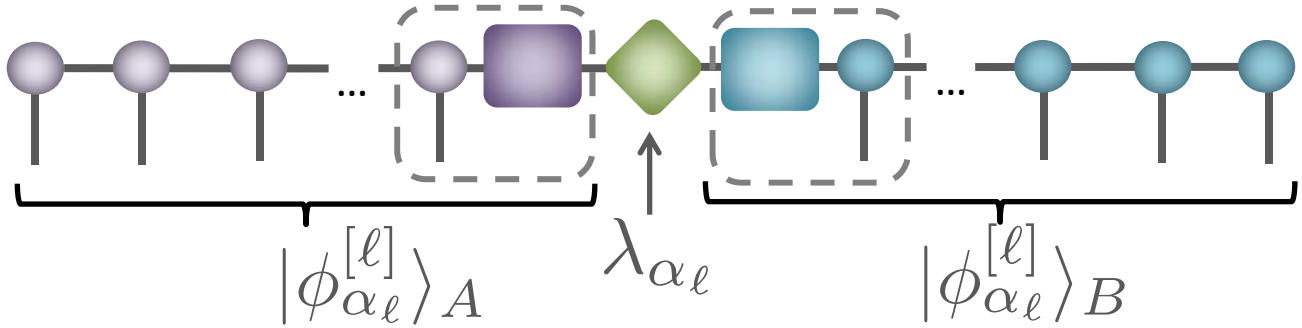
Do analogously from the right. Meet up at desired splitting:



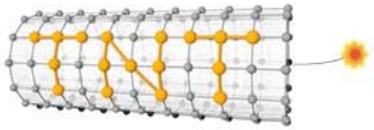
Finally we SVD the remainder:

Can absorb unitaries into adjacent **A** matrices – doesn't alter their orthonormality properties – end up with a diagonal at the split:

$$|\psi\rangle = \sum_{\alpha_\ell} \lambda_{\alpha_\ell} |\phi^{[\ell]}_{\alpha_\ell}\rangle_A |\phi^{[\ell]}_{\alpha_\ell}\rangle_B$$

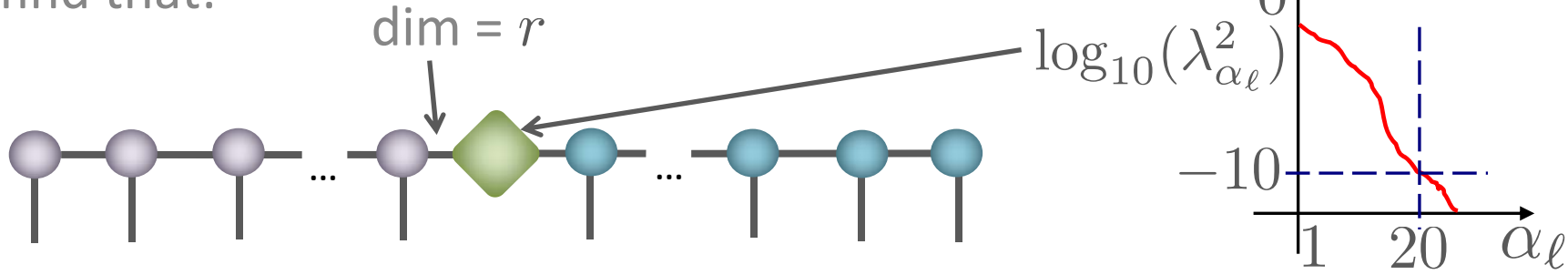


Thus we have converted the MPS into Schmidt form about the split and exposed the Schmidt coefficients (entanglement) there.



Truncation of bond

So far we have used the gauge freedom to alter the **A** matrices of the MPS – we have not altered the state represented. However what if we find that:

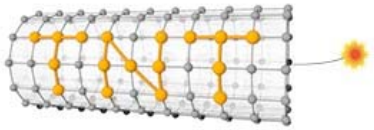


Being in Schmidt form we can identify irrelevant states, e.g. Schmidt states with a weight $\lambda^2_{\alpha_\ell} \sim 10^{-10}$, and truncate them away.

Orthogonality of the left and right states means is crucial for the optimality of this "local" truncation in terms of the global 2-norm:

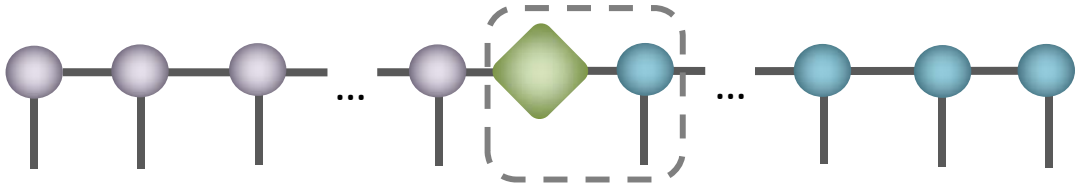
$$|||\psi\rangle - |\psi_\chi\rangle||_2^2 \leq \sum_{\alpha_\ell=\chi+1}^{r} (\lambda_{\alpha_\ell})^2$$

We have therefore "compressed" the original MPS on this single internal bond into a smaller one with very little loss of fidelity.



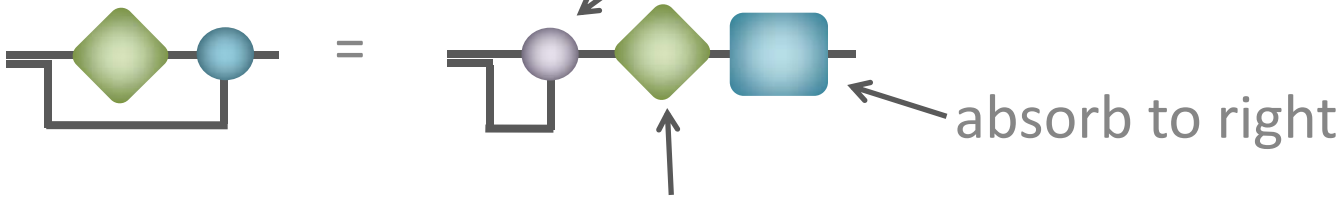
Shifting the split

Having truncated one bond how do we move to another? Simple, do another SVD to shift the splitting:



now left orthonormalised

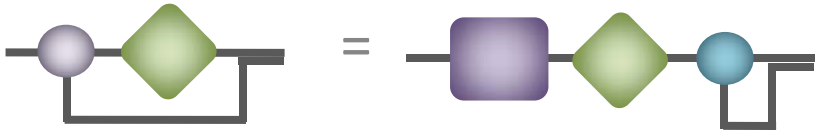
move right:



absorb to right

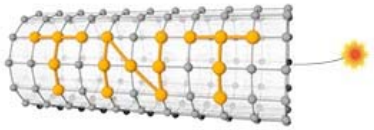
Schmidt coefficients of next bond $\lambda_{\alpha_{\ell+1}}$

move left:



analogous in this direction

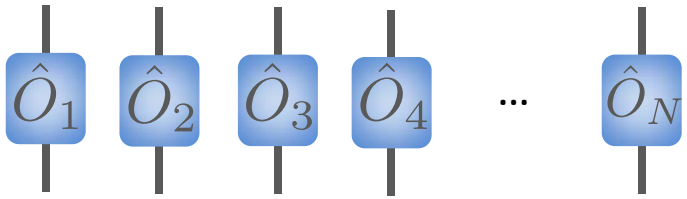
We can of course perform truncation on every bond in sequence like this. However, repeated truncations are **no** longer globally optimal.



Matrix Product Operators

The matrix product representation can be applied to operators as well as states. This will be very useful in what is to follow.

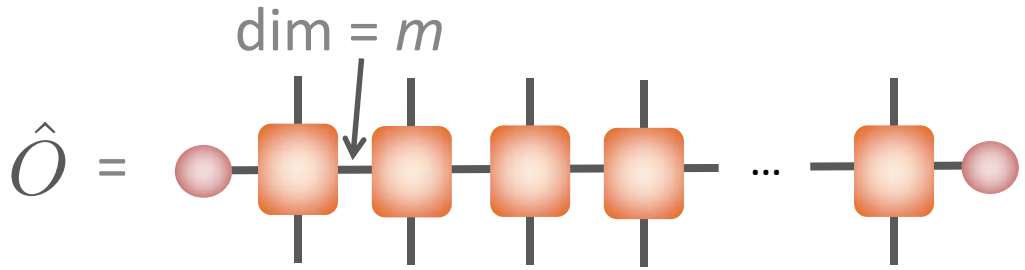
We have already encountered the simplest MPO's, product operators:



includes on-site observables, terms in Hamiltonians and *n*-point correlations:

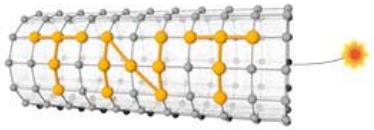
$$\hat{O}_1 \otimes \hat{O}_2 \otimes \hat{O}_3 \otimes \cdots \otimes \hat{O}_N$$

Again we generalise this by introducing internal legs:



Formally it is just an expansion in the physical basis as usual:

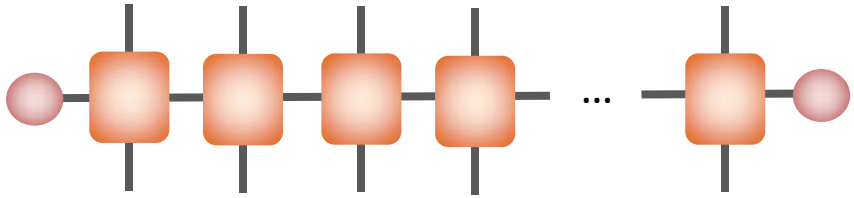
$$\hat{O} = \sum_{\vec{\sigma},\vec{\sigma}'} \vec{L}^{\dagger} \mathbf{A}^{\sigma_1 \sigma_1'} \mathbf{A}'^{\sigma_2 \sigma_2'} \cdots \mathbf{A}^{\sigma_N \sigma_N'} \vec{R} |\vec{\sigma}\rangle\langle\vec{\sigma}'|$$



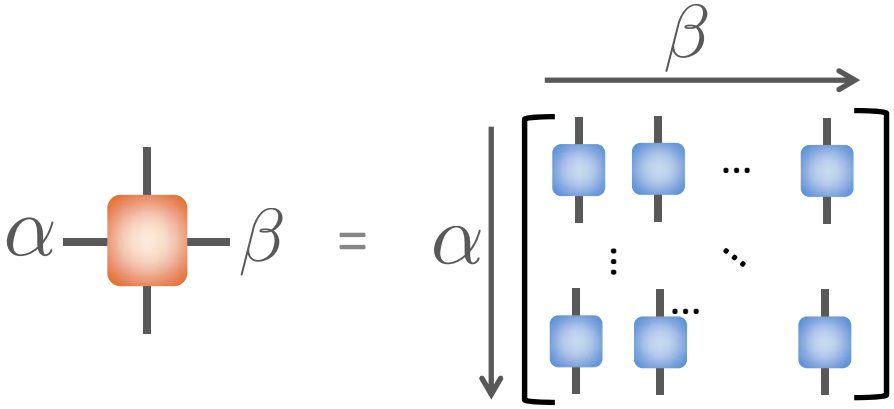
Matrix Product Operators

Many useful operators have an MPO representation with a very small internal dimension. We can build them by hand using a trick from earlier …

$\hat{O}$ =



wrap up physical legs so **A** matrix elements are on-site operators:

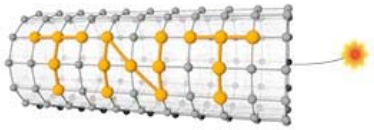


reducing the MPO to the product:

$$\hat{O} = \vec{L}^{\dagger}\mathbf{A}\mathbf{A}\cdots\mathbf{A}\vec{R}$$

and then use a lower-triangular form by choosing the *m*-dimensional boundary vectors to be:

$$\vec{L}^{T} = [0, 0, \ldots, 0, 1] \quad \vec{R}^{T} = [1, 0, \ldots, 0, 0]$$



Matrix Product Operators

Examples: Choose **A** matrices with *m* = 2 as:
Multiplication of **A** matrices gives tensor products the operators as:

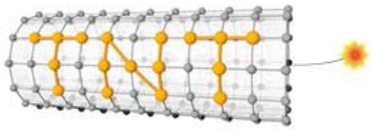
$$\mathbf{A} = \begin{pmatrix} \hat{p} & 0 \\ \hat{q} & \hat{r} \end{pmatrix}$$

$$\mathbf{A} \times \mathbf{A} = \begin{pmatrix} \hat{p} \otimes \hat{p} & 0 \\ \hat{q} \otimes p + \hat{r} \otimes q & \hat{r} \otimes \hat{r} \end{pmatrix}$$

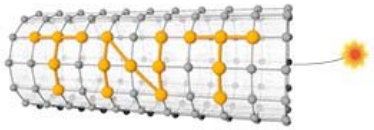
For longer products the bottom left corner becomes a sum of all translates of terms like $\hat{r} \otimes \cdots \otimes \hat{r} \otimes \hat{q} \otimes \hat{p} \otimes \cdots \otimes \hat{p}$
obvious example is … but generalises to larger *m* easily:

$$\hat{Z} = \sum_{j=1}^{N} \sigma_j^z \qquad \hat{H}_{\mathrm{IM}} = \sum_{j=1}^{N-1} \sigma_j^x \sigma_{j+1}^x + B \sum_{j=1}^{N} \sigma_j^z$$

$$\mathbf{A} = \begin{pmatrix} \mathbb{1} & 0 \\ \sigma^z & \mathbb{1} \end{pmatrix} \qquad \mathbf{A} = \begin{pmatrix} \mathbb{1} & 0 & 0 \\ \sigma^x & 0 & 0 \\ B\sigma^z & \sigma^x & \mathbb{1} \end{pmatrix}$$



2: Variational optimisation

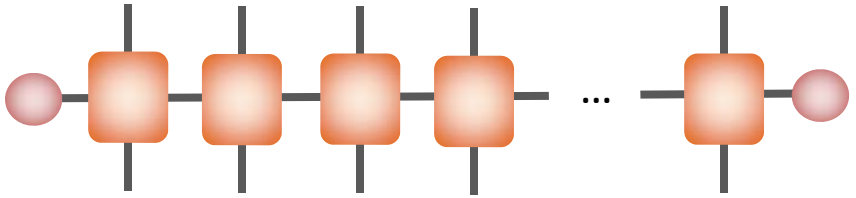


Finding the ground state

We have seen how we can compute physical properties efficiently if given an MPS. We now come to the variational calculation of an MPS approximation of the ground state …

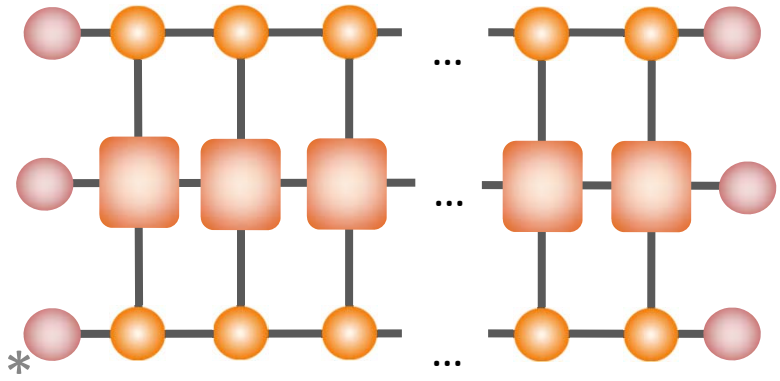
Our task is to minimise: $\epsilon = \dfrac{\langle\Psi|\hat{H}|\Psi\rangle}{\langle\Psi|\Psi\rangle}$ with $|\Psi\rangle \in \mathrm{MPS}(\chi)$

We can construct an MPO for $\hat{H}$ =

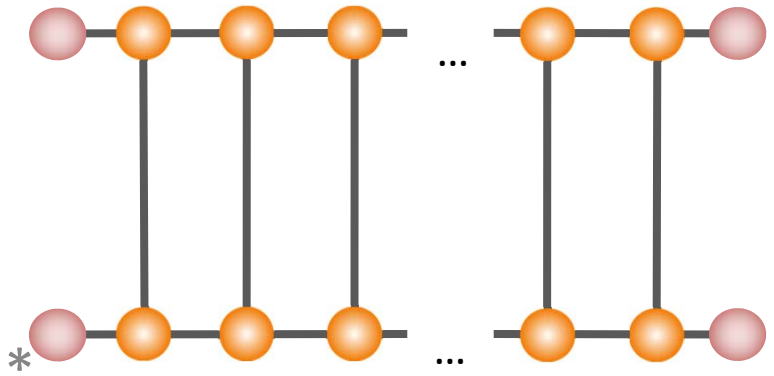


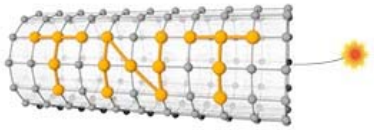
Then:

$\langle\Psi|\hat{H}|\Psi\rangle$ =



$\langle\Psi|\Psi\rangle$ =

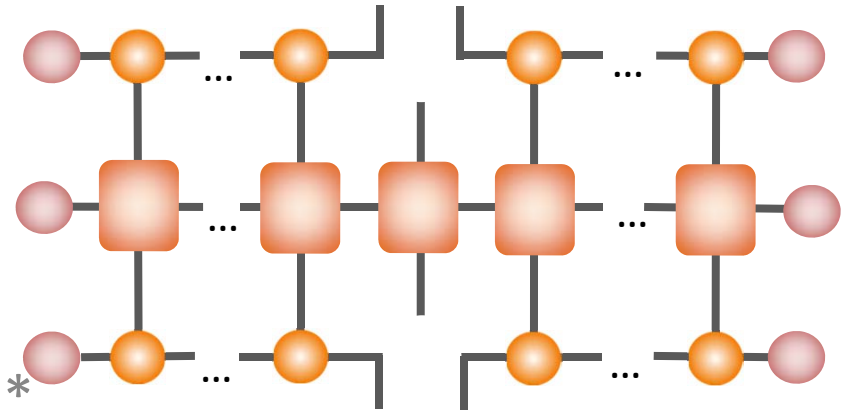




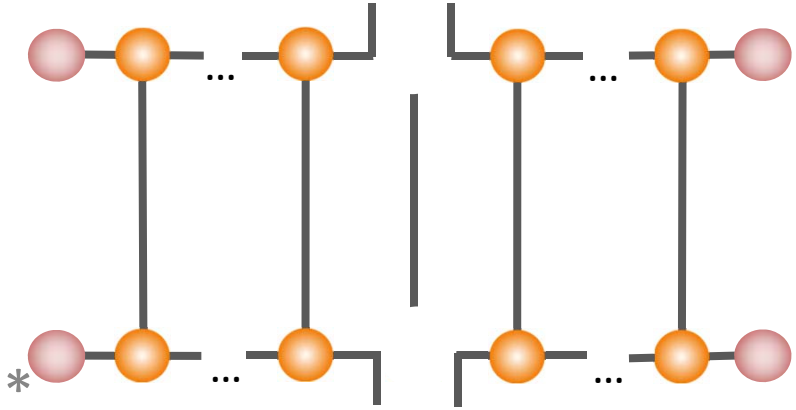
Finding the ground state

Minimising over all **A** tensors simultaneously would be very difficult. We adopt a site by site strategy instead: freeze all **A** tensors but one site's and solve a local $(d\chi^2) \times (d\chi^2)$ optimisation problem:

Effective Hamiltonian:



Effective Norm:

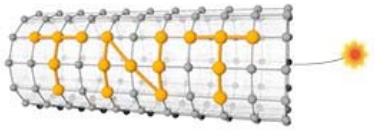


$\mathbf{H}$ =

$\vec{A}$ =

$\mathbf{N}$ =

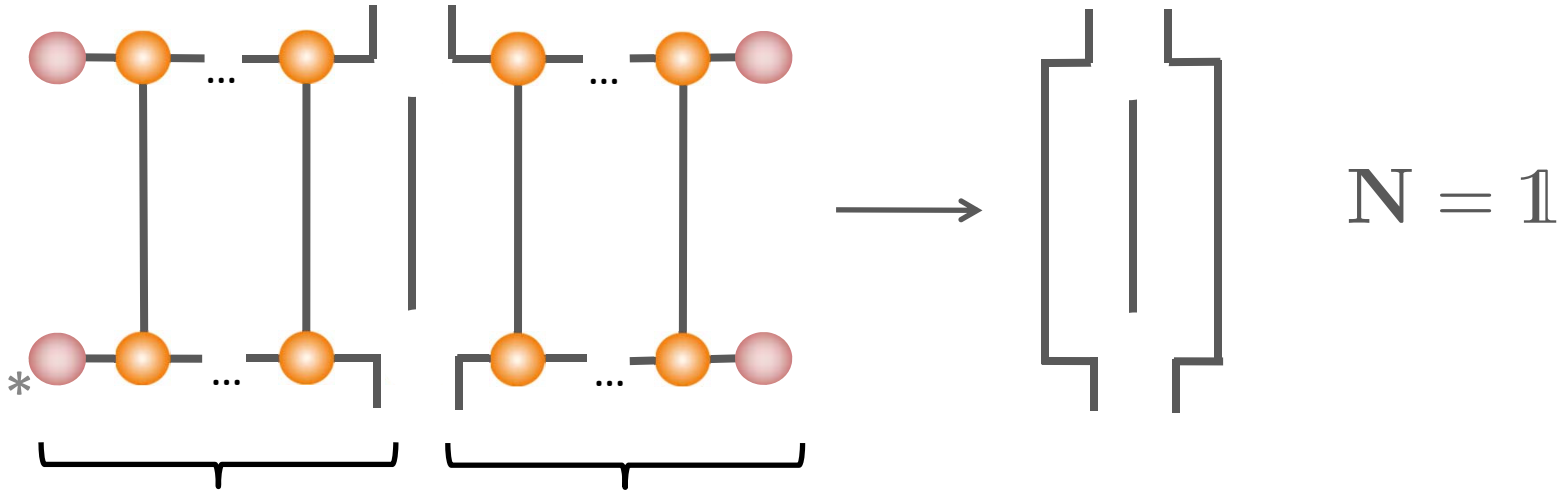
Local minimisation of $\epsilon(A) = \dfrac{\vec{A}^\dagger \mathbf{H} \vec{A}}{\vec{A}^\dagger \mathbf{N} \vec{A}} \longrightarrow \mathbf{H}\vec{x} = \lambda \mathbf{N}\vec{x}$



Finding the ground state

However, we can avoid a generalised eigenvalue problem if we use an orthonormal gauge for the MPS with a split at the site so:

Effective Norm:

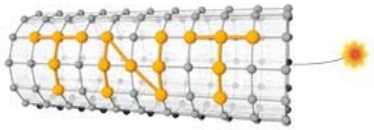


$\mathbf{N} = \mathbb{1}$

left orthonormalised right orthonormalised

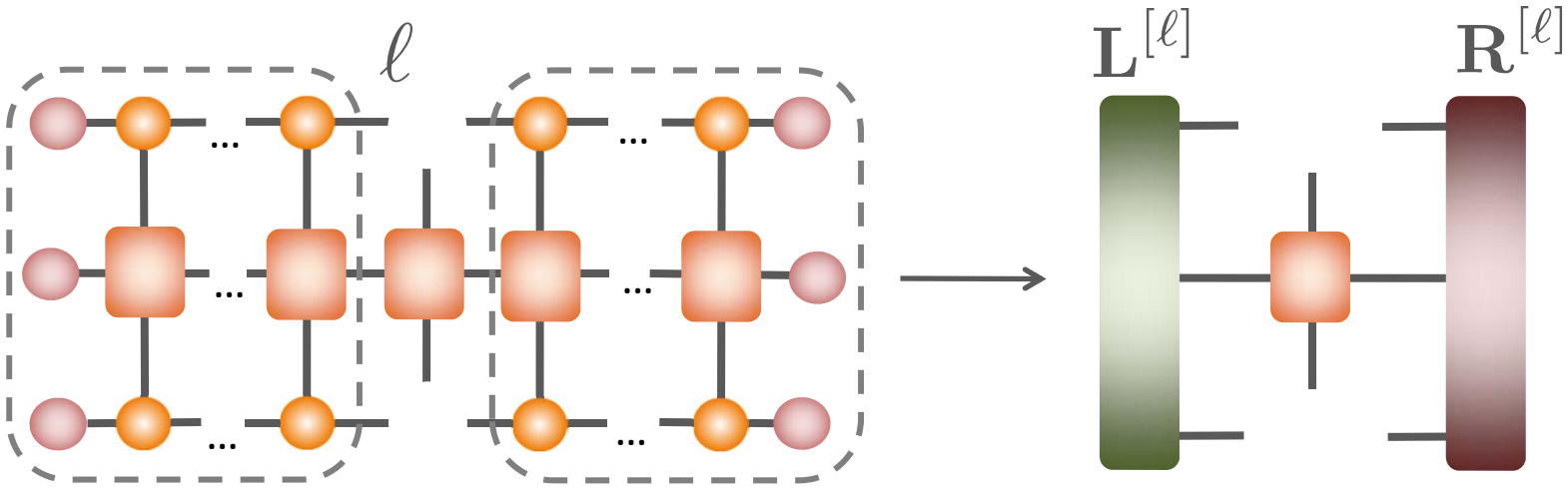
We then solve the local standard eigenvalue problem $\mathbf{H}\vec{x} = \lambda\vec{x}$

Find the lowest eigenvector and replace **A** tensor by this:

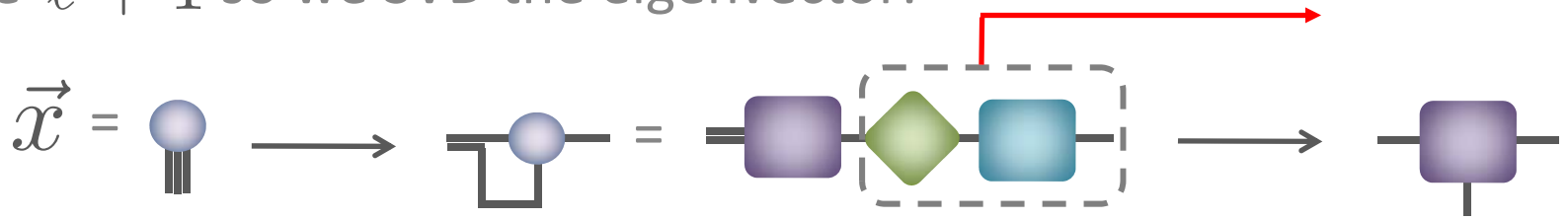


Finding the ground state

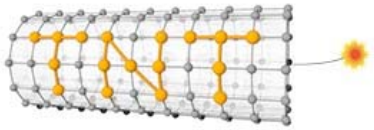
To devise a proper algorithm we need to define some intermediate tensors for this calculation. At site ℓ we have:



Given that we have minimised for site ℓ then we want to move on to site $\ell + 1$ so we SVD the eigenvector:

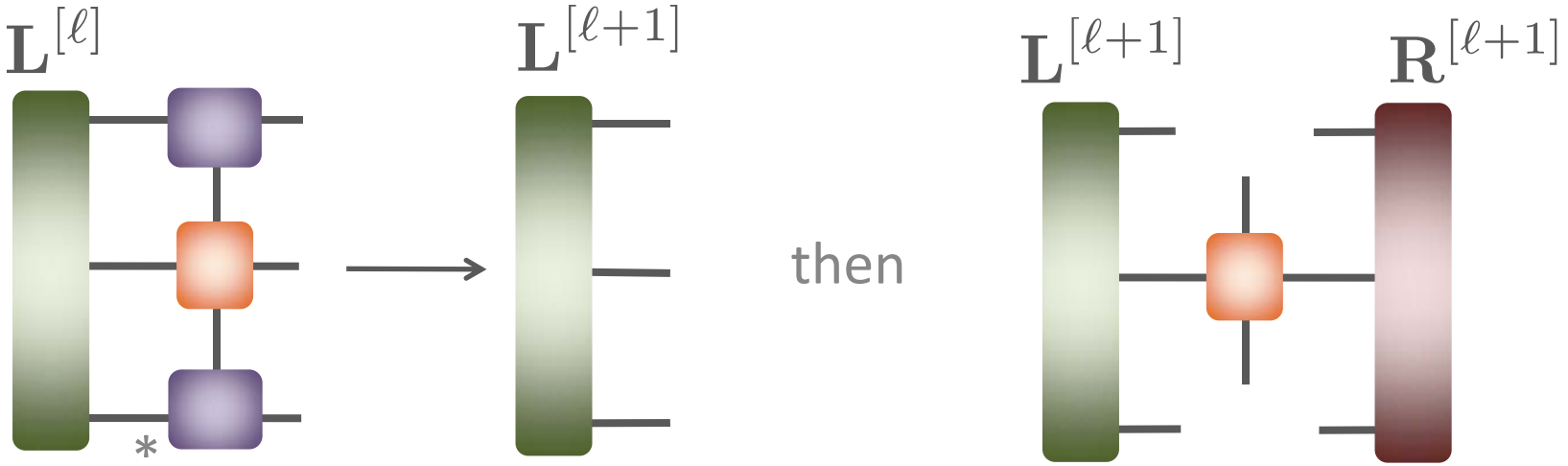


Keep the left unitary as the new **A** tensor (so its left orthonormalised), pass on the rest to improve initial guess for next **A** tensor …



Finding the ground state

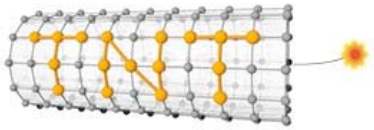
Now create a new left environment tensor one site further on:



We compute (or retrieve) the right environment tensor $\mathbf{R}^{[\ell+1]}$ and then form a new local eigenvalue problem for site $\ell + 1$

This algorithm thus performs a left => right alternating least-squares minimisation sweep of the **A** tensors individually.

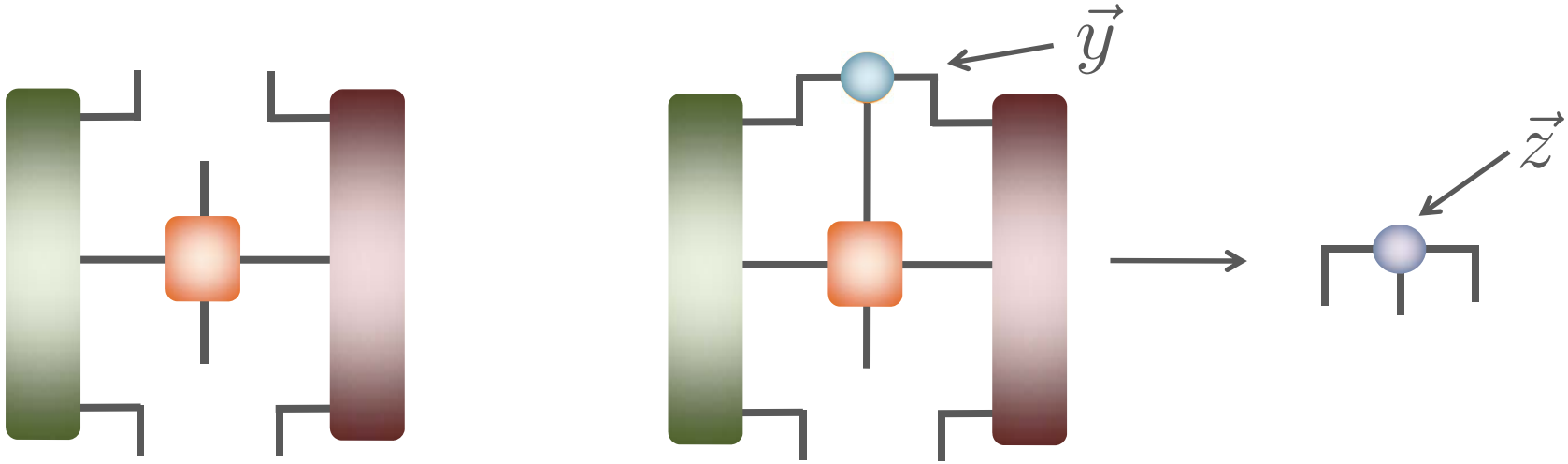
The right => left sweep is defined analogously. Overall algorithm consists of repeated left=>right and right=>left sweeps until convergence in energy (usually only a handful are needed).



Finding the ground state

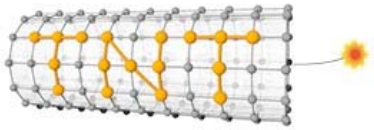
What was the cost?

Each local eigenvalue problem formally scales as $\mathcal{O}(d^3\chi^6)$ if solved completely. However, we only want the lowest eigenvector – use iterative methods (Jacobi-Davidson or Lanczos) instead …



In fact we should not even build $\mathbf{H}$ explicitly since iterative solvers only need to know the matrix-vector multiplication $\mathbf{H}\vec{y} = \vec{z}$.

This operation costs $\mathcal{O}(d\chi^3 m)$ after exploiting internal structure.



TNTgo example

Let's try an example DMRG calculation using our online web interface tool:

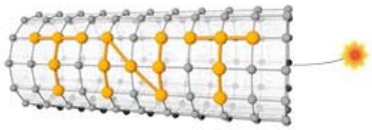
www.tntgo.org

We can compute the ground state for the isotropic Heisenberg Hamiltonian for some OBC chain of length *N:*

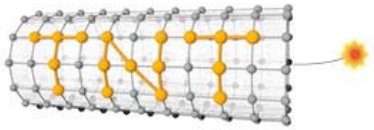
$$\hat{H} = J \sum_{j=1}^{N-1} \left(\hat{S}_j^x \hat{S}_{j+1}^x + \hat{S}_j^y \hat{S}_{j+1}^y + \hat{S}_j^z \hat{S}_{j+1}^z \right)$$

The exact energy density in the thermodynamic limit, given by the Bethe ansatz solution is:

$$E_0/J = -0.43929075655$$

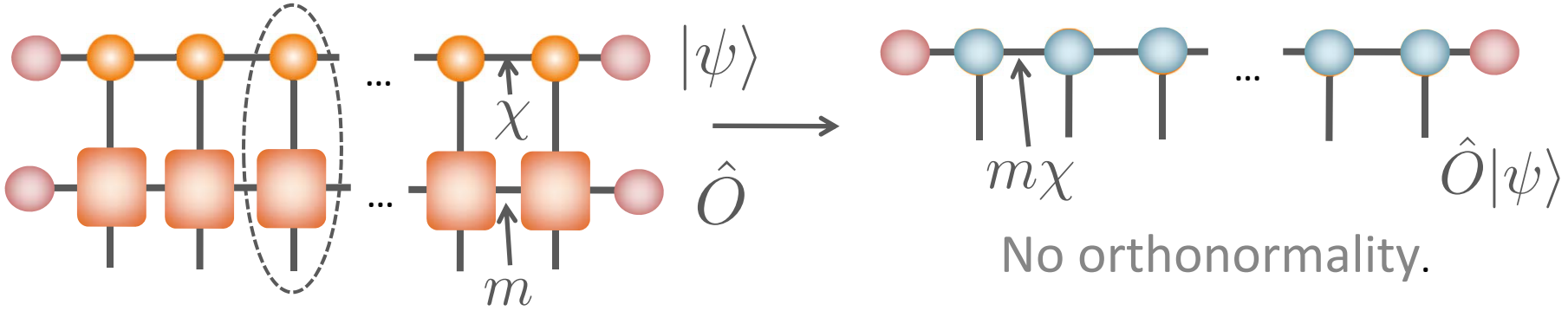


3: Time-evolution



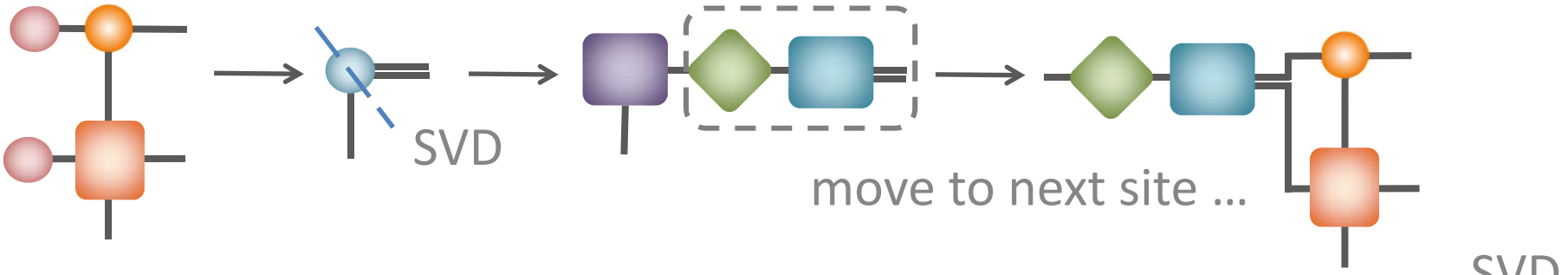
MPO x MPS zip

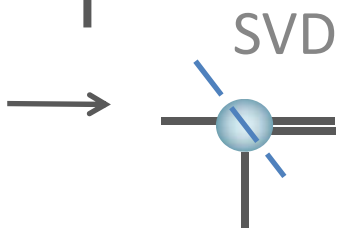
Suppose we have an MPO and we wish to apply this to an MPS and get the MPS for the resulting state. Exact approach is to contract …



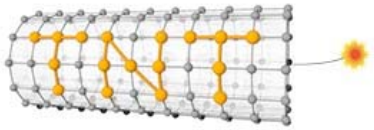
No orthonormality.

The MPS dimension grows exponentially with the number of MPOs we multiplied by. Can we approximate/compress? Yes, use SVD again:



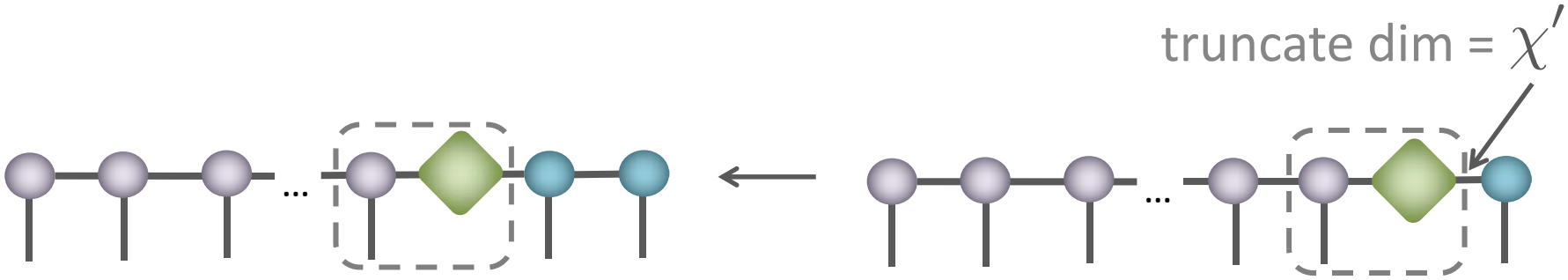


Contract and SVD, sweeping from left to right, passing on the remainder, but don't truncate yet.

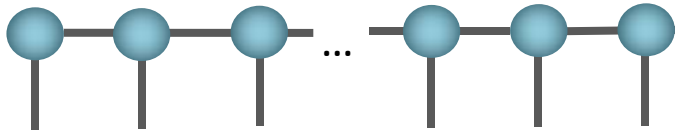


MPO x MPS zip

We will establish a fully left orthonormalised state, but with an enlarged internal dimension $m\chi$. Now we sweep back and truncate:

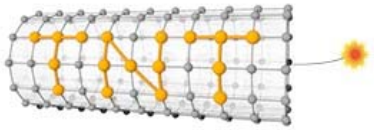


Orthonormality ensures locally optimal truncation, but not globally. This is because of the one-sided interdependence of truncations. Continuing we end up with a fully right orthonormalised MPS with dimension χ'...



$$\hat{O}|\psi\rangle \rightarrow |\psi'\rangle$$

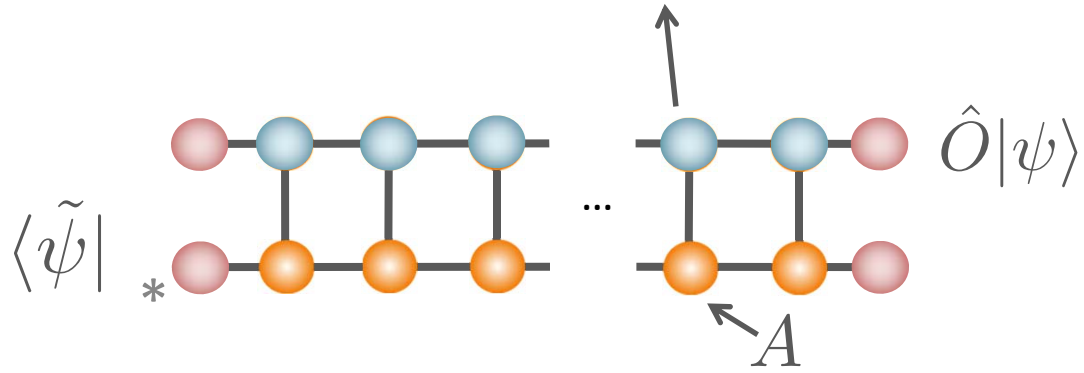
Whether compression is accurate depends on the Schmidt spectra encountered during the sweep. Overall the cost is $\mathcal{O}(m^3\chi^3)$.



Variational MPO x MPS

An alternative approach is to variationally target the 2-norm residual between the "exact" MPO x MPS and a compressed MPS result:

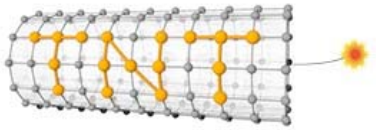
$$||\hat{O}|\psi\rangle - |\tilde{\psi}\rangle||_2^2 = \langle\psi|\hat{O}^\dagger\hat{O}|\psi\rangle - \langle\tilde{\psi}|\hat{O}|\psi\rangle - \langle\psi|\hat{O}^\dagger|\tilde{\psi}\rangle + \langle\tilde{\psi}|\tilde{\psi}\rangle$$



This is a highly non-linear optimisation problem in terms of the **A** matrices of the compressed MPS. Use the strategy of extremising with respect to one **A** matrix at a time (will all the others frozen) …

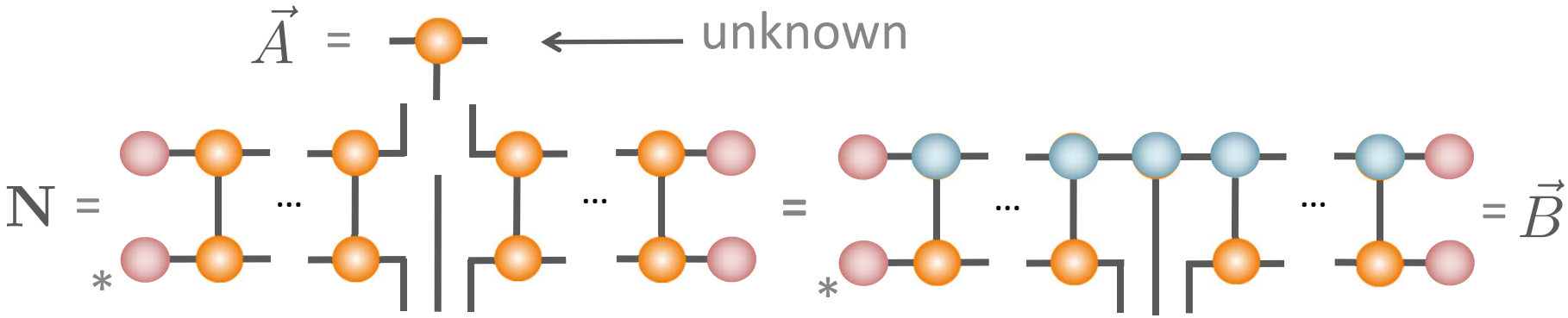
$$\min_A ||\hat{O}|\psi\rangle - |\tilde{\psi}\rangle||_2^2 = \min_A \left(\vec{A}^\dagger \mathbf{N}\vec{A} - \vec{A}^\dagger\vec{B} - \vec{B}^\dagger\vec{A} + \vec{B}^\dagger\vec{B}\right)$$

Solution found from a linear system of equations: $\mathbf{N}\vec{A} = \vec{B}$



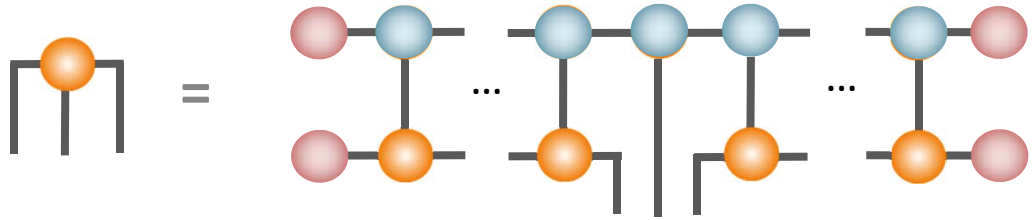
Variational MPO x MPS

Graphically the equation $\mathbf{N}\vec{A} = \vec{B}$ is equivalent to:

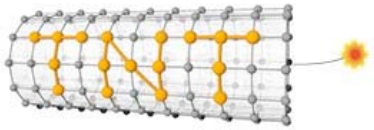


Can be solved iteratively, e.g. by using conjugate gradients, where the solution $\mathbf{N}^{-1}\vec{B}$ is found by using repeatedly the multiplication $\mathbf{N}\vec{A}$. However, if left-right orthonormality applies to the frozen **A** matrices then the problem reduces to …

No system of equations to solve now!



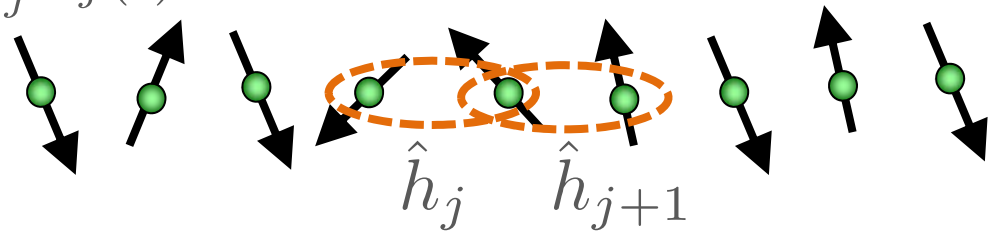
Use SVD on local solution to shift orthonormality split one site along. Efficiency depends heavily on the initial guess for the **A** matrices.



Time-evolution of MPS

We have seen how to efficiently compute physical properties from MPS and how to efficiently find a variational approximation to the ground state, now we come to how to efficiently time-evolve MPS …

Take a Hamiltonian $\hat{H}(t) = \sum_j \hat{h}_j(t)$ composed of time-dependent nearest-neighbour terms, e.g.

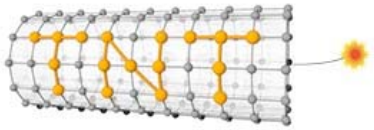


Given some initial MPS $|\psi\rangle$ at time t_0 we want to compute the action of the unitary time-evolution operator:

$$\hat{U}(t, t_0) = \mathbb{T}\left\{ e^{-i \int_{t_0}^{t} \hat{H}(s) \mathrm{d}s} \right\}$$

which formally solves to the TDSE $i \frac{\mathrm{d}}{\mathrm{d}t} |\psi(t)\rangle = \hat{H}(t) |\psi(t)\rangle$ as:

$$|\psi(t)\rangle = \hat{U}(t, t_0) |\psi\rangle$$

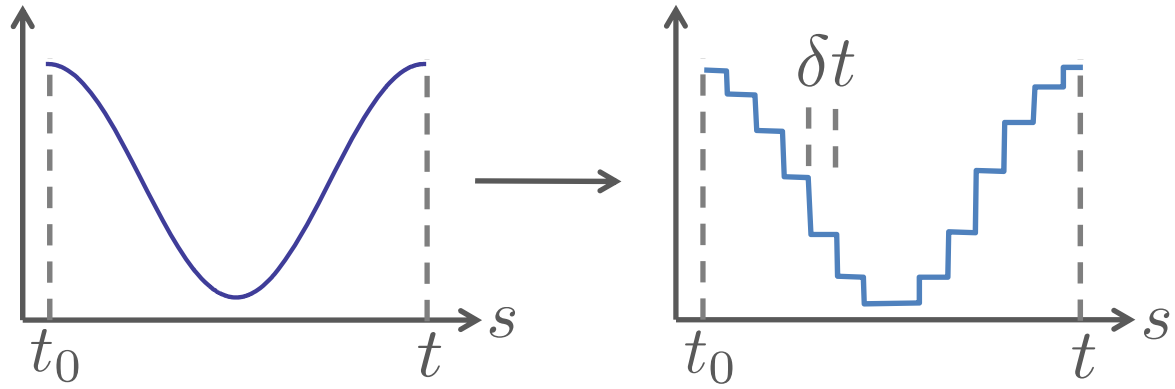


Time-evolution of MPS

To handle this we first digitise the time-dependence into ***T*** piece-wise constant segments δt:

$$\hat{U}(t,t_0) = \prod_{k=1}^{T} \hat{U}_k$$

$$\hat{U}_k = e^{-i\hat{H}(k)\delta t}$$

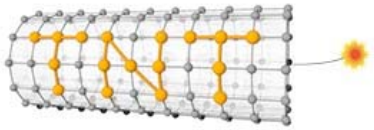


However, we are still left with an exponentially sized unitary $\hat{U}_k$ for any given segment, so next step is to "Trotterise". Simplest case:

$$e^{(\hat{A}+\hat{B})x} = e^{\hat{A}x}e^{\hat{B}x} + \mathcal{O}(x^2)$$

akin to assuming $\hat{A}$ and $\hat{B}$ commute (not true), so usually more accurate higher-order versions are used like:

$$e^{(\hat{A}+\hat{B})x} = e^{\hat{A}x/2}e^{\hat{B}x}e^{\hat{A}x/2} + \mathcal{O}(x^3)$$

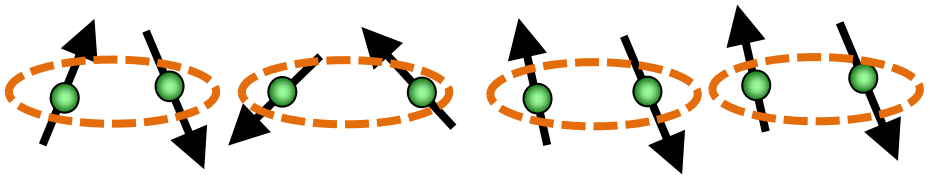


Time-evolution of MPS

For a given segment we have: $\hat{U}_k = e^{-i\delta t \sum_j \hat{h}_j(k)}$

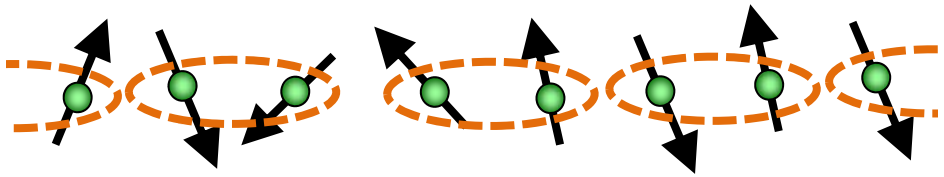
we then divide the terms in the Hamiltonian into two parts:

even pairs:



$$\hat{A} = \sum_{j \in \text{even}} \hat{h}_j(k)$$

odd pairs:

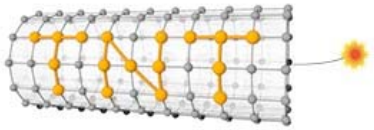


$$\hat{B} = \sum_{j \in \text{odd}} \hat{h}_j(k)$$

Notice that (for spins and bosons) all terms **within** either set *commute* since they act on disjoint sites, so for example:

$$e^{-i\hat{B}\delta t} = \exp\left(-i\delta t \sum_{j \in \text{odd}} \hat{h}_j(k)\right) = \prod_{j \in \text{odd}} \exp\left(-i\delta t\ \hat{h}_j(k)\right)$$

is an **exact** expansion and is simply a product of two-site unitaries.

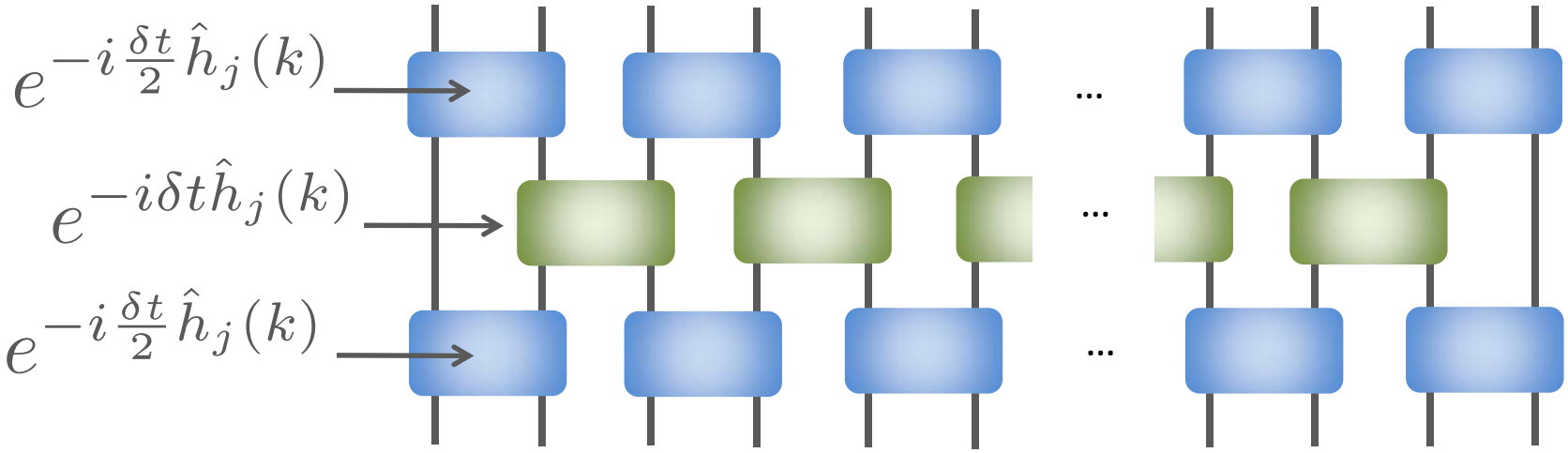


Time-evolution of MPS

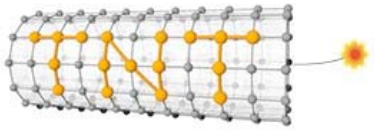
The evolution for a single segment can be approximated to $\mathcal{O}(\delta t^3)$ as:

$$\hat{U}_k = \prod_{j\in\text{even}} \exp\left(-i\frac{\delta t}{2}\,\hat{h}_j(k)\right) \prod_{j\in\text{odd}} \exp\left(-i\delta t\,\hat{h}_j(k)\right) \prod_{j\in\text{even}} \exp\left(-i\frac{\delta t}{2}\hat{h}_j(k)\right)$$

This is equivalent to applying a quantum circuit of two-site unitaries:

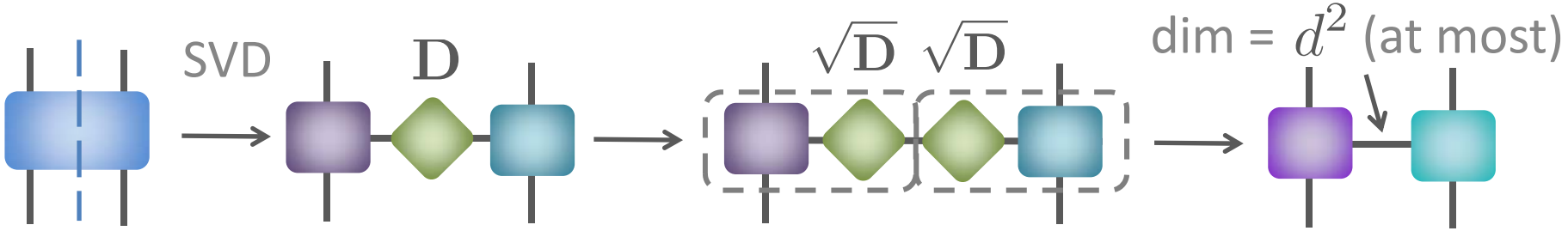


Other segments will be analogously decomposed for their times. We need $\epsilon\delta t \ll 1$ where ϵ is a relevant energy scale, both for smoothly approximating time-variations and reducing Trotter errors.

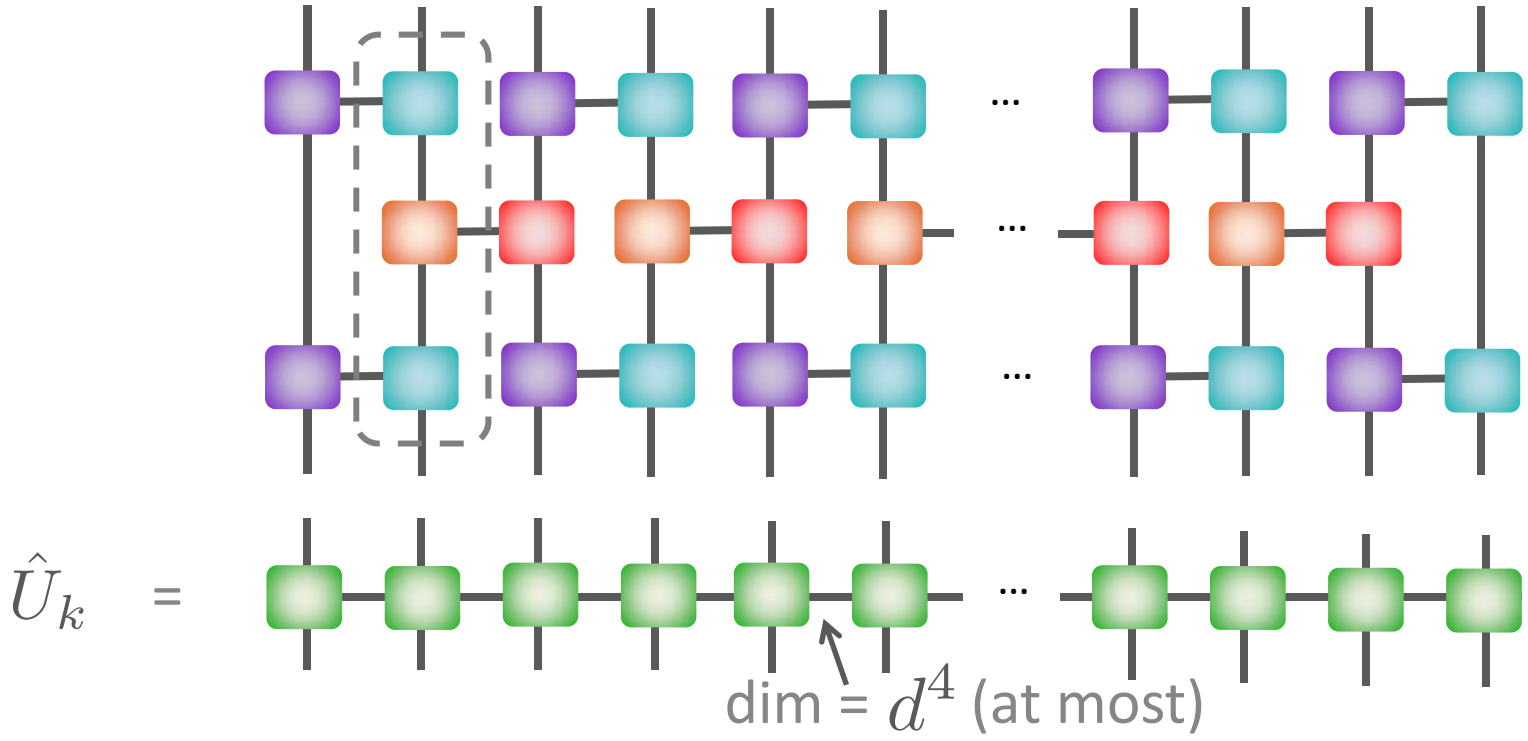


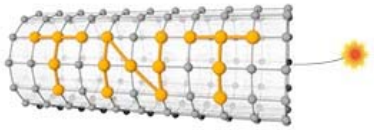
Time-evolution of MPS

We now recast this circuit as an MPO. First we SVD the gates like:



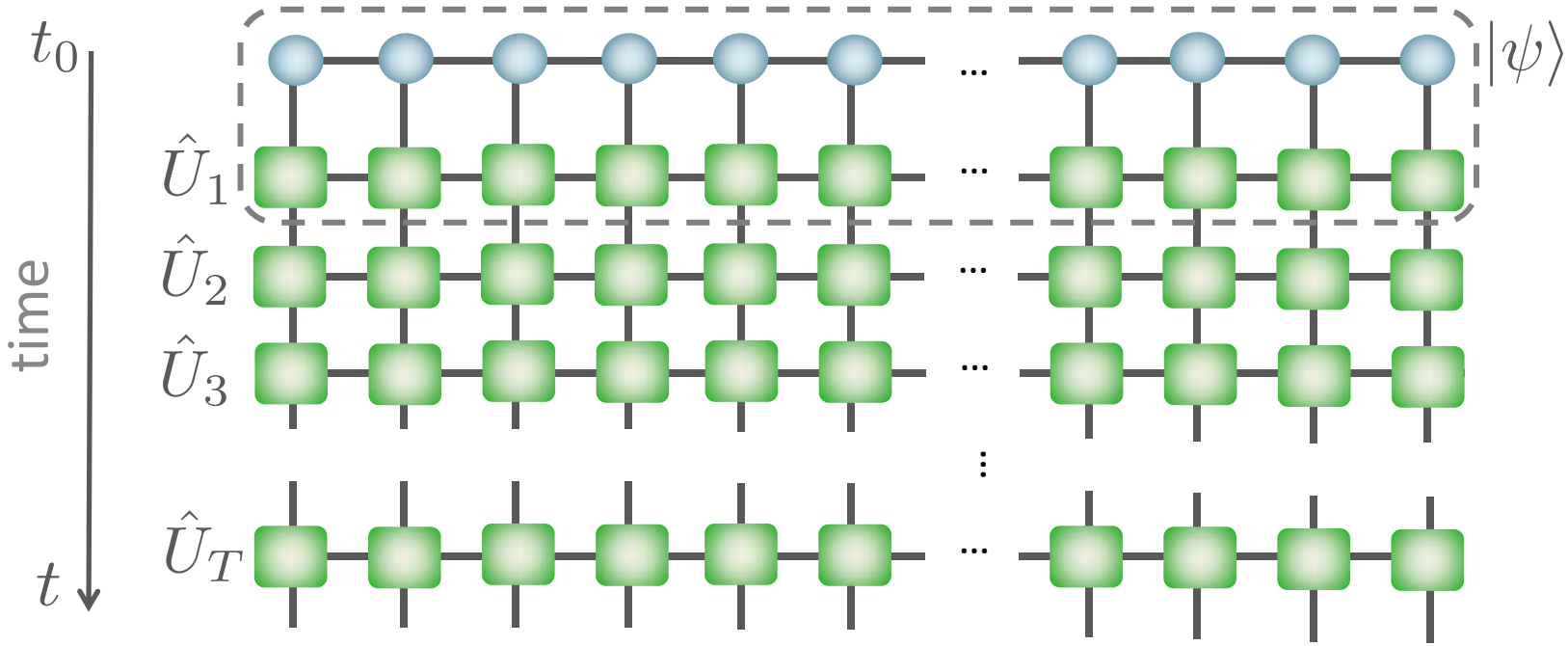
Then insert into circuit and contract vertically to get a segment MPO:



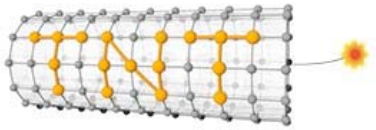


Time-evolution of MPS

The complete time-evolutions is now an MPS repeatedly multiplied by a sequence of MPOs for each time segment:



The t-MPS algorithm proceeds by starting at the top and performing one by one each of the MPO x MPS for each row of the grid, while compressing the resulting MPS to control its internal dimension.



Calculations we can do

In this lecture we have presented efficient algorithms for solving problems (1) finding the GS and (2) doing time-evolution for 1D strongly correlated systems using MPS. As examples we can:

- Find a ground state in some regime of a Hamiltonian.
- Apply an excitation to it (like a spin-flip) and time-evolve.
- Or, time-evolve with a time-dependent Hamiltonian.

As well as straight dynamical evolution of an initial state we can also compute spectral functions by considering two evolutions:

$$|\psi(t)\rangle = \hat{U}(t,0)|\psi\rangle \quad |\psi_B(t)\rangle = \hat{U}(t,0)\hat{B}|\psi\rangle$$

then compute the time-dependent overlap of them:

$$\langle\psi|\hat{A}(t)\hat{B}(0)|\psi\rangle = \langle\psi(t)|\hat{A}|\psi_B(t)\rangle$$

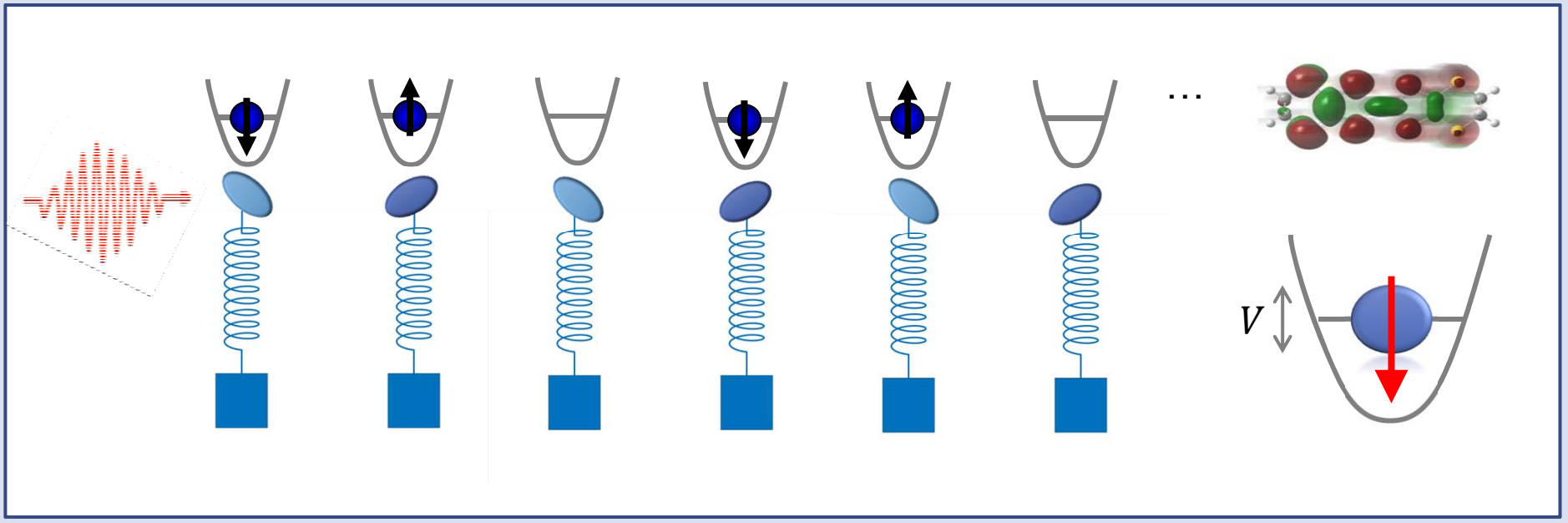
Fourier transform w.r.t. *t* to get $S_{AB}(\omega)$.

Driving local vibrations

- Include driving of a-b lattice with frequency Ω

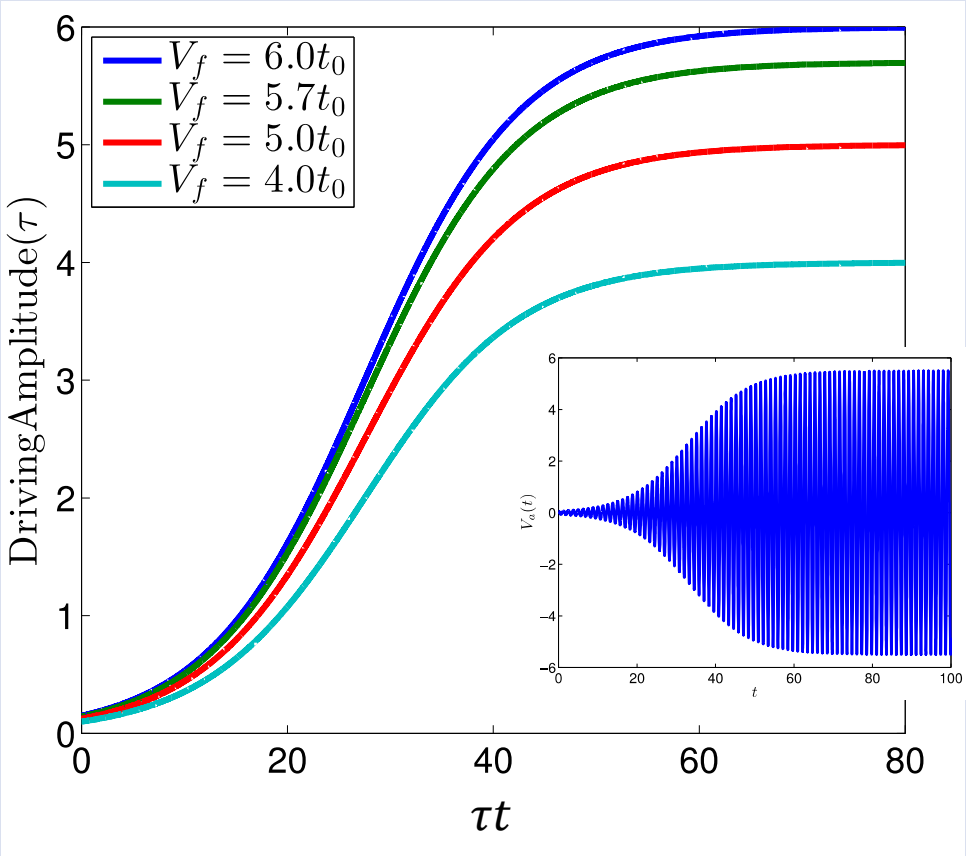
$$H_d(\tau) = \frac{V}{2}\sum_{j\in a} \sin(\Omega\tau - \phi)\, n_j + \frac{V}{2}\sum_{j\in b} \sin(\Omega\tau + \phi)\, n_j$$

we assume the driving to be out of phase $\phi = \frac{\pi}{2}$



Simulations for 1D infinite system

We use td-iDMRG to study the dynamics for a slowly increasing drive and starting from the ground state



We look at correlation functions and their structure factors

Density-density correlations

$$N_{ij} = \langle n_i n_j \rangle - \langle n_i \rangle \langle n_j \rangle$$

Spin-spin correlations

$$S_{ij} = \langle S_i^z S_j^z \rangle$$

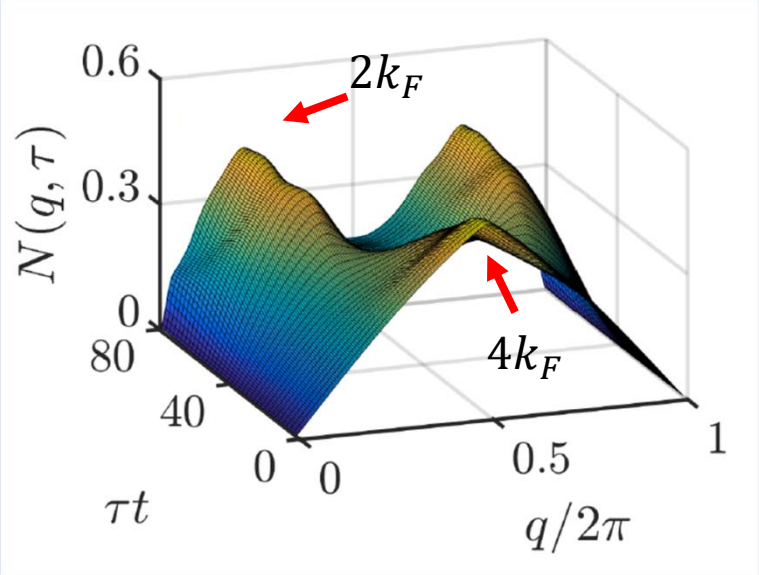
where $S_i^z = (n_{i\uparrow} - n_{i\downarrow})/2$

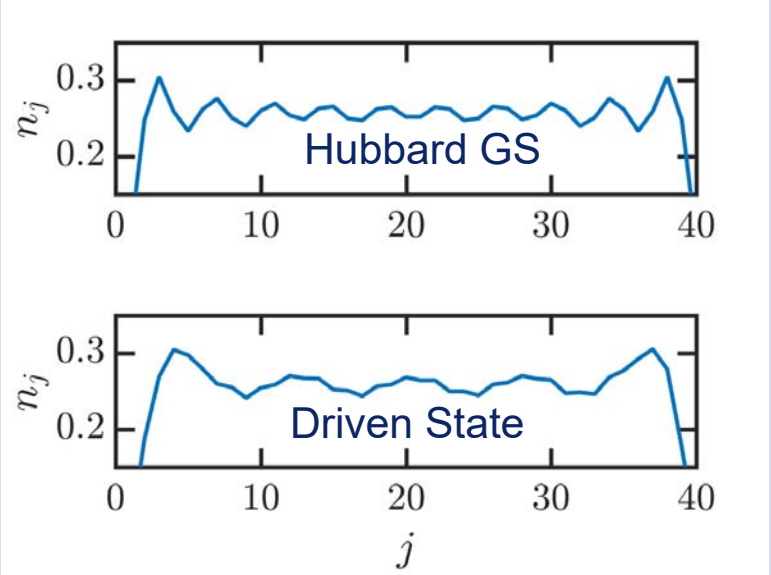
Pair correlations

$$P_{ij} = \langle b_{i,i+1}^{\dagger}\, b_{j,j+1} \rangle$$

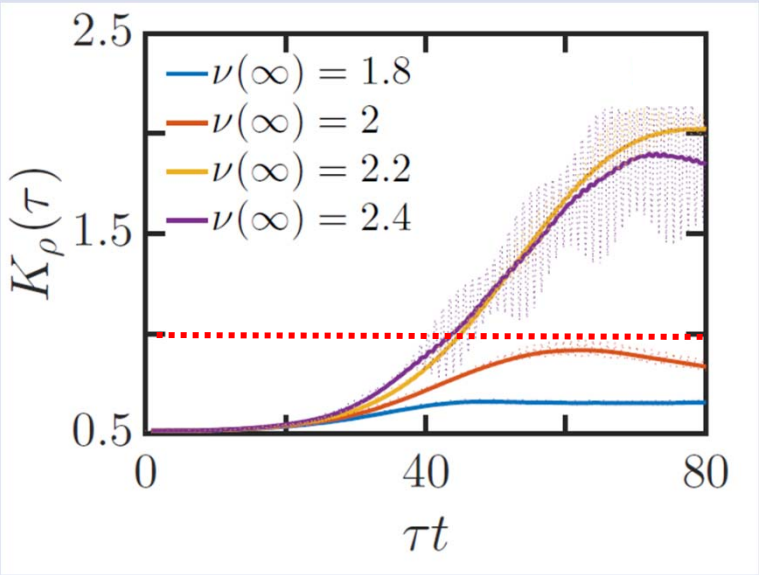
where $b_{ij} = (c_{i\uparrow} c_{j\downarrow} - c_{i\downarrow} c_{j\uparrow})/\sqrt{2}$

Density structure factor





$U = 20t$
$\Omega = 6t$
$L = \infty$
$\bar{n} = 1/2$
$k_F = \pi/4$

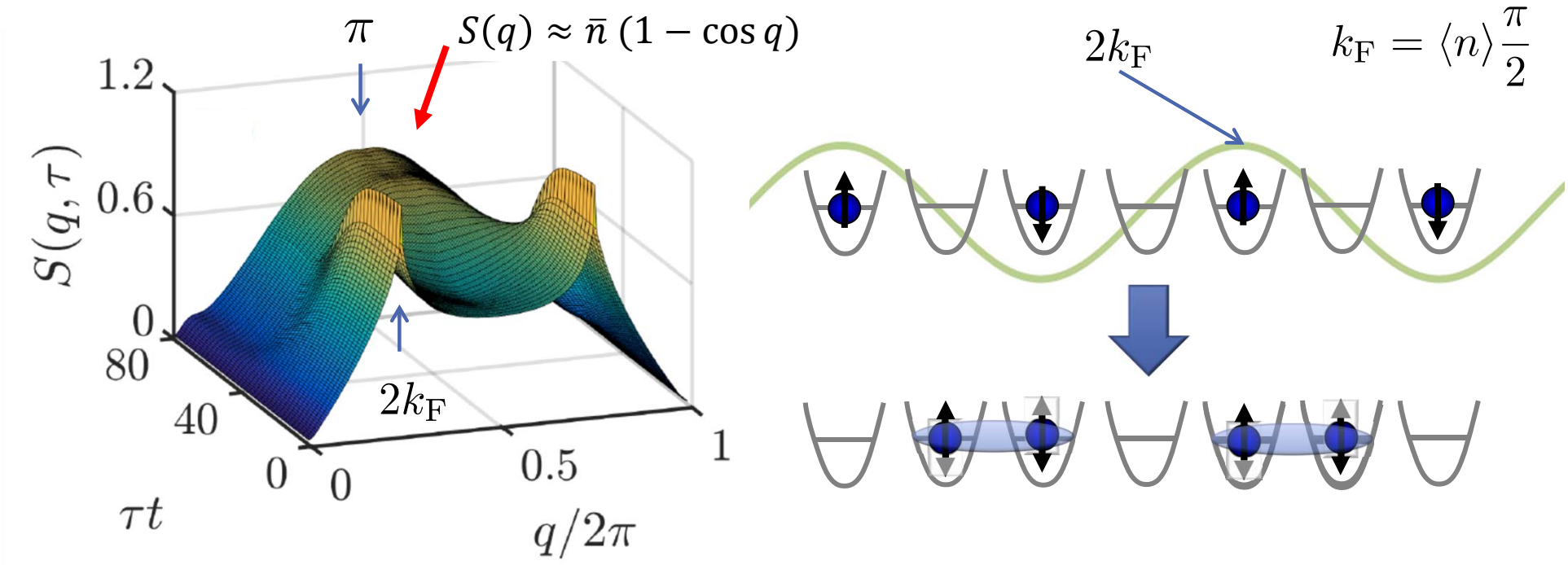


For small quasi-momentum q

$$N(q) \approx \frac{K_\rho q}{\pi}$$

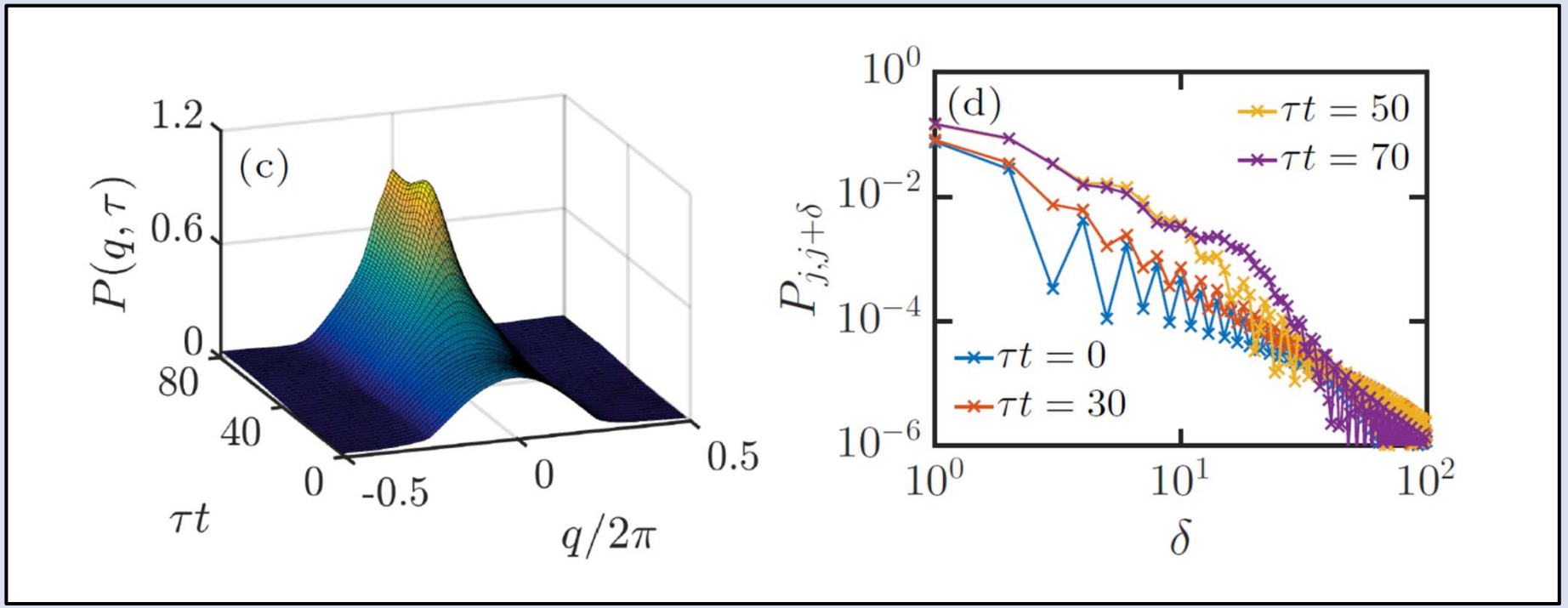
Repulsive Luttinger liquid (M) for $K_\rho < 1$
Attractive Luttinger liquid (SC) for $K_\rho > 1$

Spin structure factor



The $q = 2k_F$ spin wave in the Hubbard ground state gives way to AFM bound pairs.

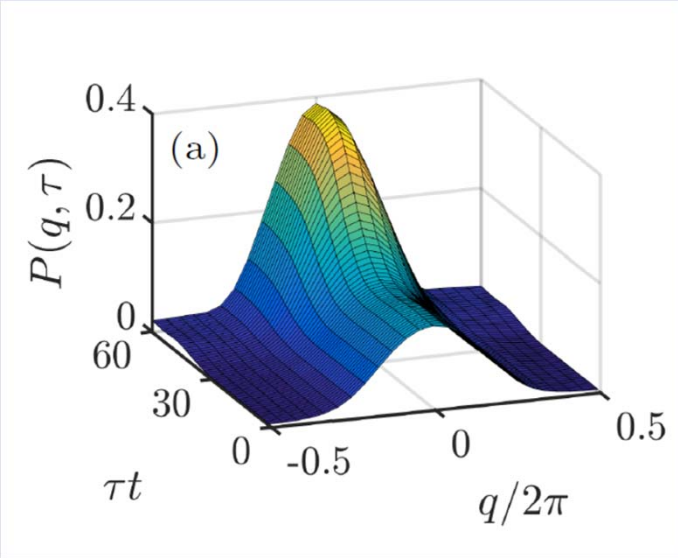
Pair correlations

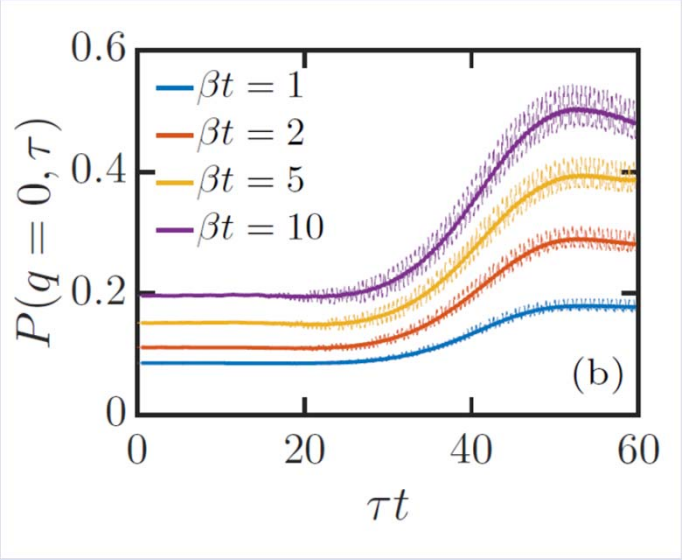


Long-range pair correlations are enhanced by the driving

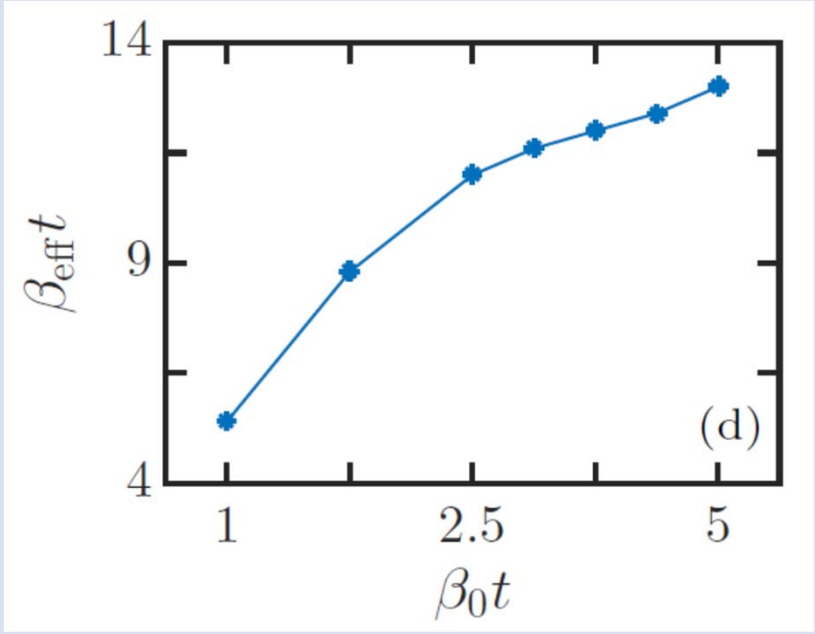
Correlations spread with speed $t/4 \approx J$

Finite temperature β_0 driven state

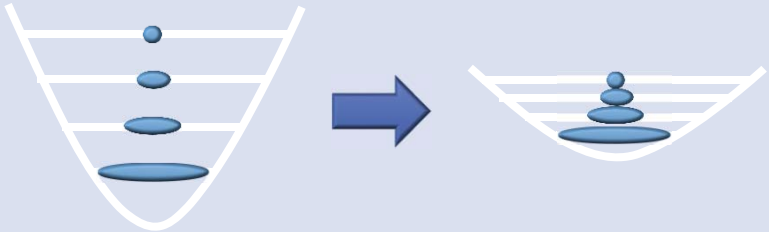


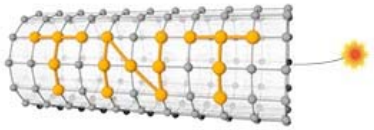


Effective inverse temperature β_{eff} of the driven state in the tJ model



Many-body version of adiabatic cooling





Code for doing TNT

You could write your own, but our group is developing an open-access TNT library (in C) which does a lot of the hard work for you:

www.tensornetworktheory.org

Available:

- DMRG and td-DMRG available now.
- U(1) quantum number symmetry.

Coming soon:

- Finite-temperature calculations.
- Master equation evolution.
- Quantum trajectory code.

Coming later:

- Parallelised versions of codes.
- Impurity solvers for DMFT.
- Tensor tree, PEPS and MERA.