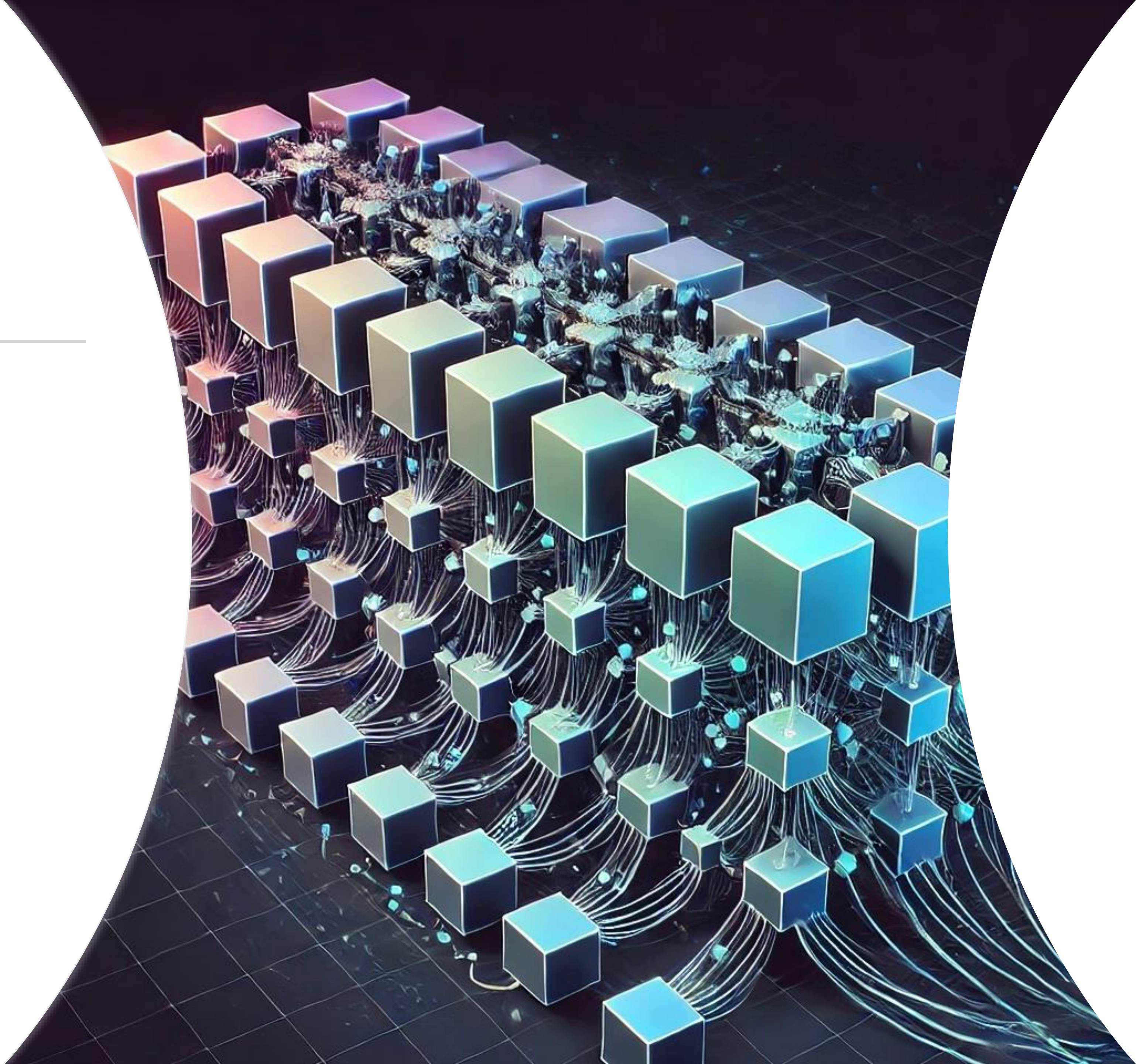
Quantic Tensor Train (QTT)

Scaffidi’s Group meeting Oct 22nd

Omid Tavakol



Outline:

- How to call a function that is represented by MPS?
- How to integrate a function using MPS?
- How to make MPS that generates a desired function?
- Examples

Most of the talk is based on:

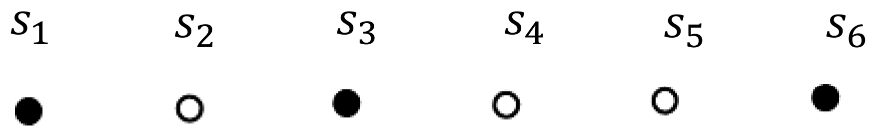
Tensor network for machine learning applications I,II,II

Learning Feynman Diagrams with Tensor Trains

Multiscale Space-Time Ansatz for Correlation Functions of Quantum Systems Based on Quantics Tensor Trains

Quantum picture:

Let say we have N qubits. These qubits can generate 2^Nconfiguration. Each configuration can represent a number $0 \leq x \leq 1$.



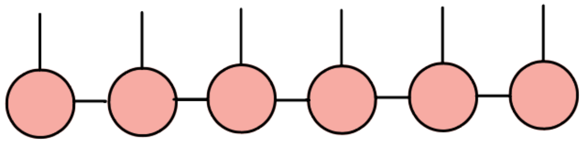
$$x(\boldsymbol{s}) = \sum_{i=1}^{N} \frac{s_i - 1}{2^i}$$

$s_i = 1,2$
Julia convention

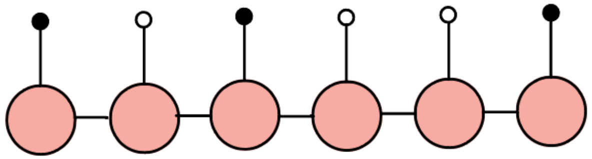
We can represent a function with a state in the Hilbert state

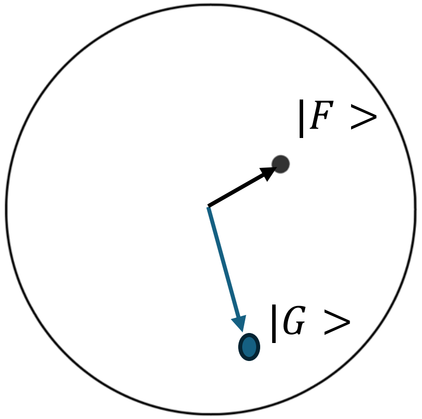
$$|F> = \sum_{s}^{2^N} f(x)|x(\boldsymbol{s})> \quad , \quad |G> = \sum_{s}^{2^N} g(x)|x(\boldsymbol{s})>$$

$|F> =$



$f(x) = \langle x|F\rangle =$



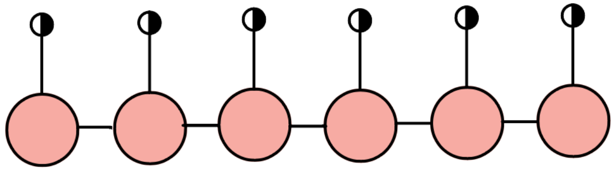


Integrating a function using MPS

Let us make the state $|\phi > = \frac{1}{2^N}\sum_{s}^{2^N} |x(s) >$. Large superposition but this is a product state $|\phi > \propto \ H^N|0 >$.

$$\langle\phi|F\rangle = \frac{1}{2^N}\sum_{s'}^{2^N}\sum_{s}^{2^N} f(x_s)\,\langle x_{s'}|x_s\rangle = \frac{1}{2^N}\sum_{s}^{2^N} f(x) \approx \int_0^1 f(x)dx$$

$\langle\phi|F\rangle =$



where

$= [\begin{smallmatrix}1/2\\1/2\end{smallmatrix}]$

Some important reminders

DMRG is an algorithm to solve Quantum Hamiltonian. How ? The dimension is 2^N

Tensor Network ansatz: Dimension reduction

How to reduce the dimension of a tensor?
1) Singular Value Decomposition (SVD)
2) Tensor Cross Interpolation (TCI)
3) Tensor Recursive Sketches (TRS)

I would like to focus more on what we can achieve with these algorithms, rather than on their implementation, since they have already been implemented.

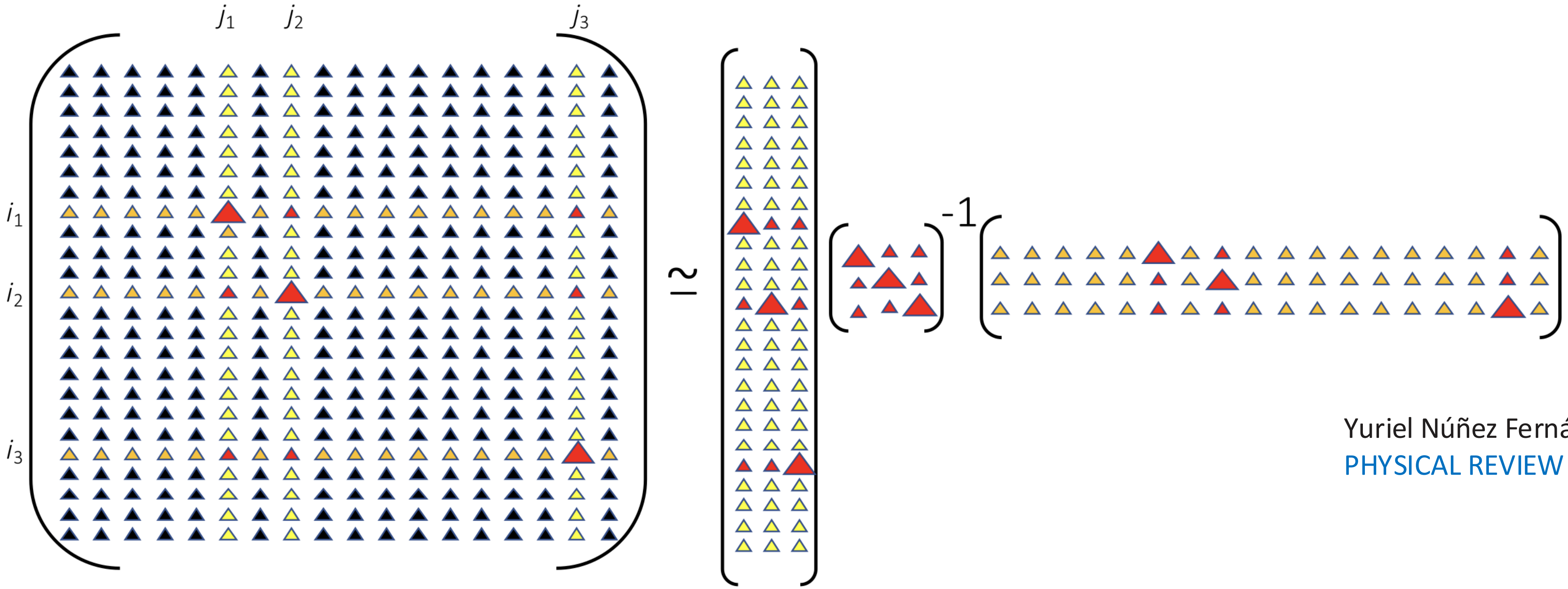
Can I always perform SVD? Technically yes, but not always in a practical way.

In principle, SVD can be applied to any tensor to convert it into an MPS (Matrix Product State). However, the bond dimension can grow significantly. SVD is particularly useful when **you have access to the full matrix**, which is a critical point to keep in mind. For instance, you can apply SVD when using an MPO on an MPS, since all the necessary tensors (or at least their approximations) are available.

Can we find the MPS representation of a tensor by knowing **a reasonable number of rows and columns** from the actual matrix? Yes, this can be achieved using the TCI (Tensor Cross Interpolation) and TRC (Tensor Recursive Sketches) algorithms.

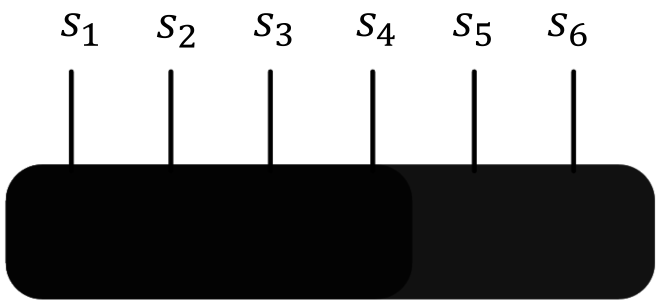
Tensor Cross Interpolation (TCI)

The idea for matrix cross interpolation:



Yuriel Núñez Fernández
PHYSICAL REVIEW X 12, 041018 (2022)

$s_i = 1,2$



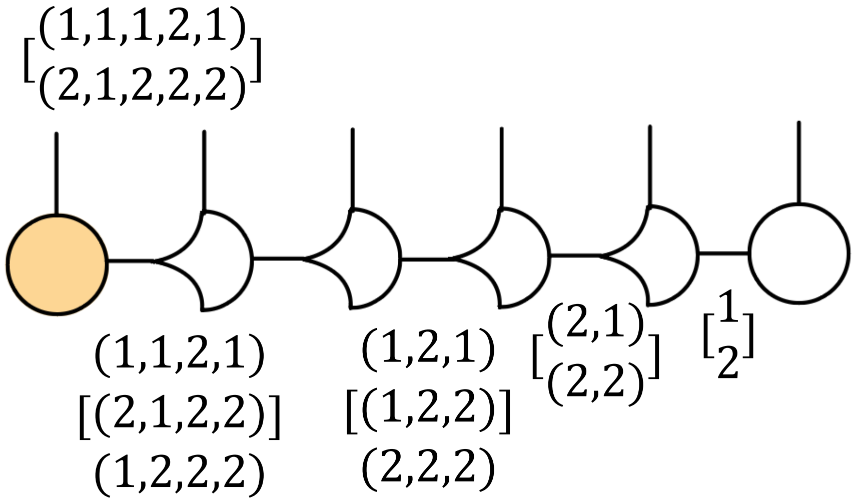
$f(1,1,1,1,2,1)$

$f(2,1,1,1,2,1)$

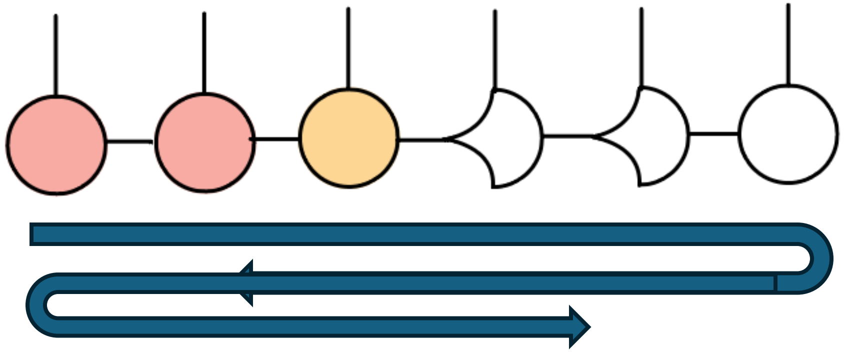
$f(1,2,1,2,2,2)$

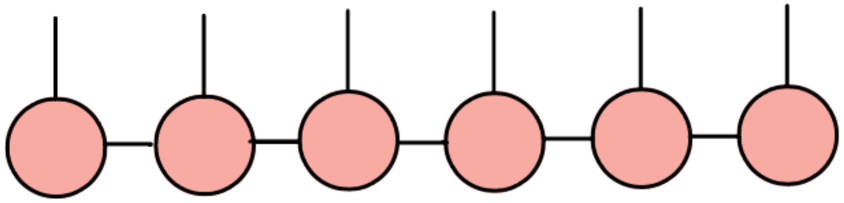
$f(2,2,1,2,2,2)$

$=$

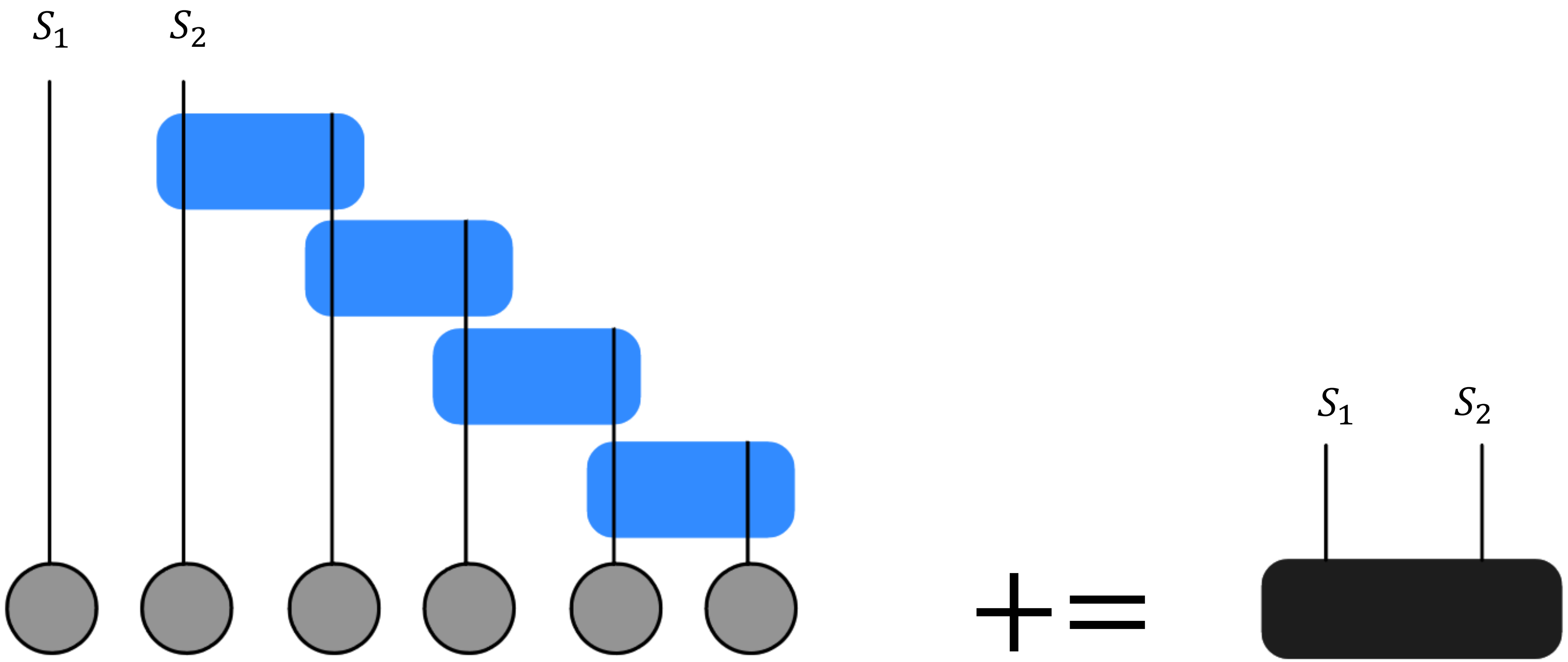


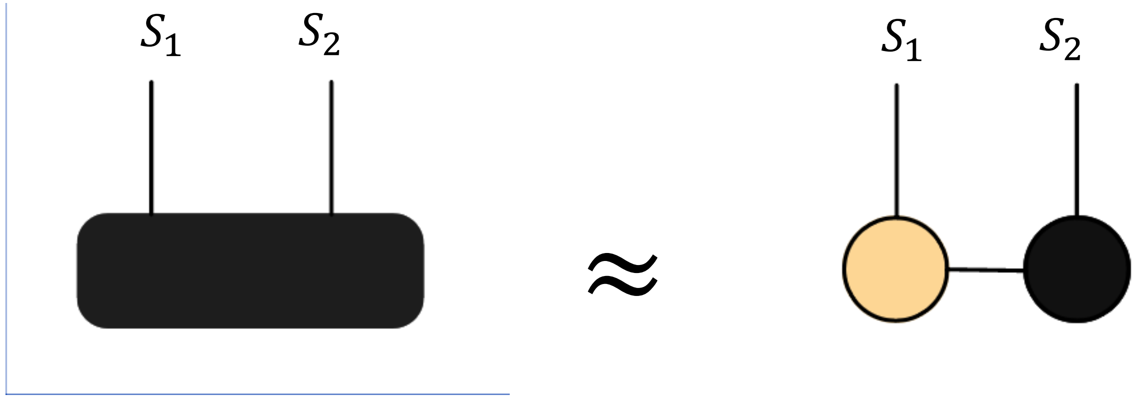
$f(s_1 f s(_{21} J s_{22}), s_3, J_3)$

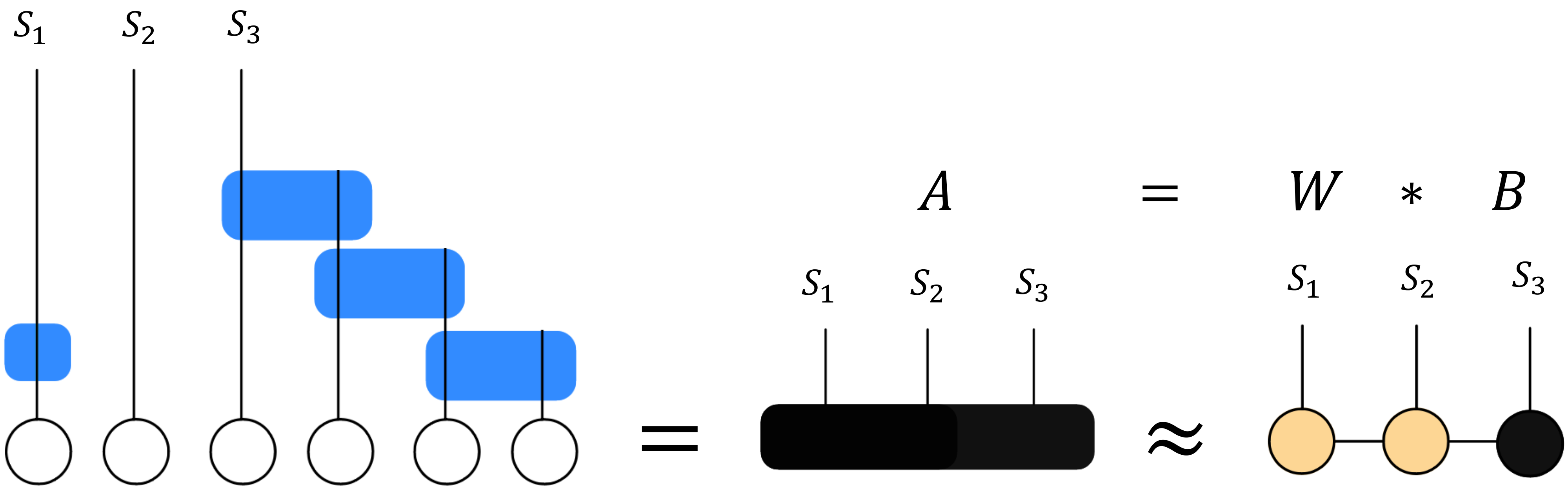




Tensor Recursive Sketches (TRS)




S_1
S_2
$\approx$
S_1
S_2


S_1
S_2
S_3
$=$
A
$=$
W
$*$
B
S_1
S_2
S_3
$\approx$
S_1
S_2
S_3

Examples

Calculating with the MPS:

1) Sum over Matsubara Frequency (different network)

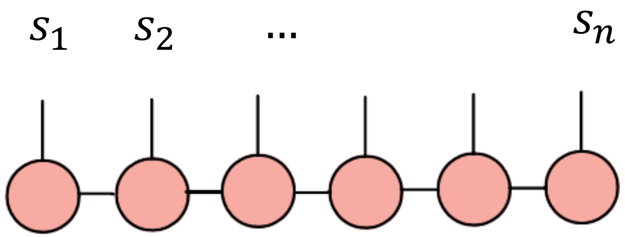
2) Polarization in 1D (Linhard function+ cos(p) as dispersion)

3) 2D Polarization Graphene

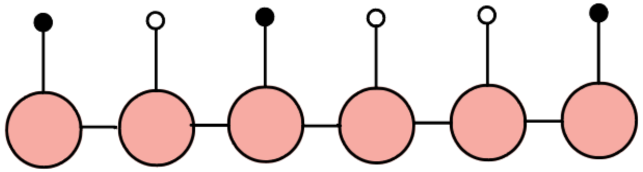
Simplest Example:

Sum over the Matsubara frequency

$G^{E,\beta}(n) =$

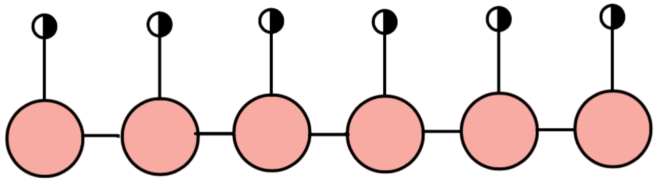


Example: Given $s = \{2,1,2,1,1,2\}$ then $n = 18$ and $i\omega_n = i(2n+1)\pi/\beta$



$$= \frac{1}{i\frac{\pi}{\beta}37 \ - E}$$

$\sum_{n=-2^L}^{2^L} G^{E,\beta}(n) =$

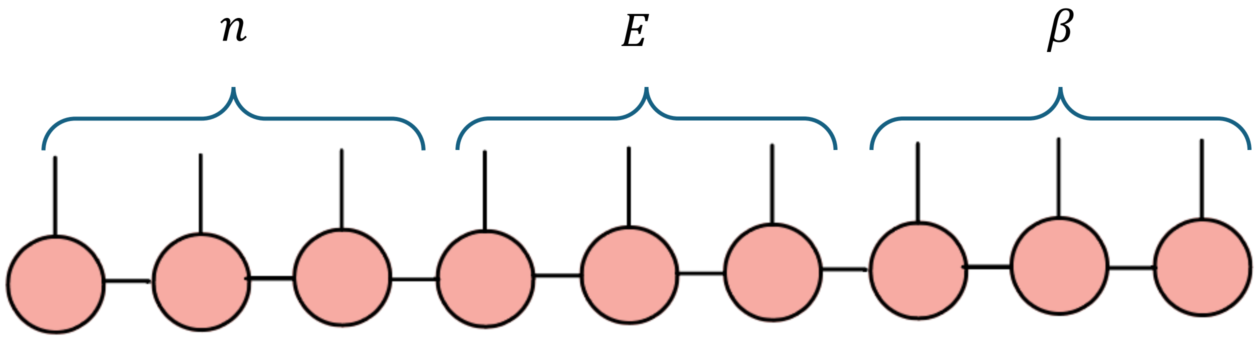


where ◑ $= \begin{bmatrix}1\\1\end{bmatrix}$

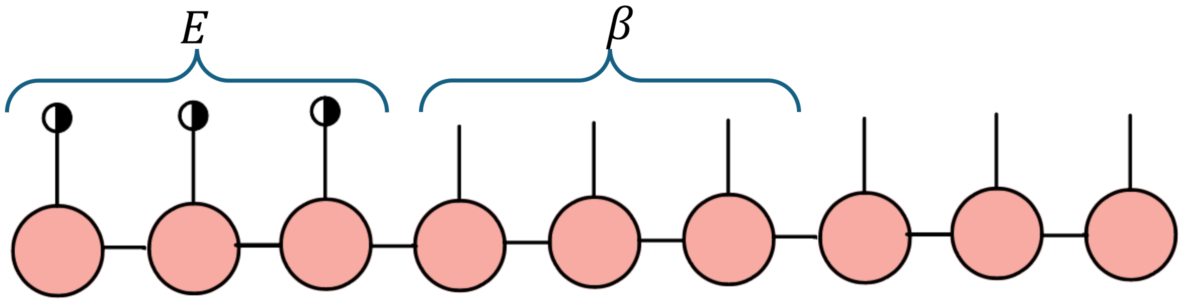
NN_QC/Tensor_train/TCI_Example_nF.ipynb Part I

Another Network:

$G(n, E, \beta) \quad =$

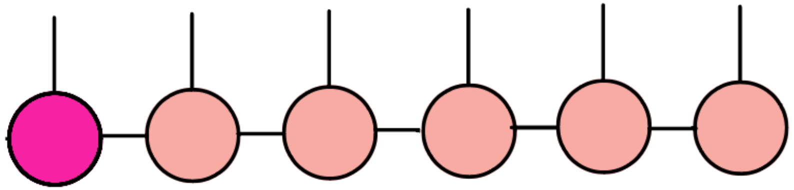


$nF(E, \beta) \quad =$



NN_QC/Tensor_train/TCI_Example_nF.ipynb Part II

$nF(E, \beta) \quad =$



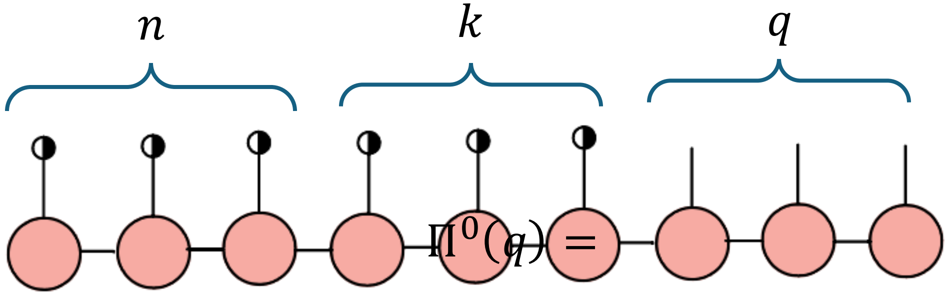
1D polarization:

$$\Pi(iv,q) = -\frac{\mathrm{T}}{2^{\mathrm{L}_k+1}} \sum_{k=-2^{L_k}}^{2^{L_k}} \sum_{n=-2^{L_n}}^{2^{L_n}} G(i\omega_n,k)G(i\omega_n+iv,k+q)$$

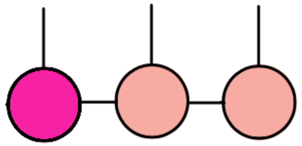
Let say we assume that $iv = 0$ thus we have an integrand as a function of $(i\omega_n, k, q)$ which we should integrate over $i\omega_n, k$

Also we need a dispersion where we consider $\epsilon(p) = p^2$

$\Pi^0(q) =$



This is what we save



Compile the code julia reload_MPS.jl 3000 (number of grids)

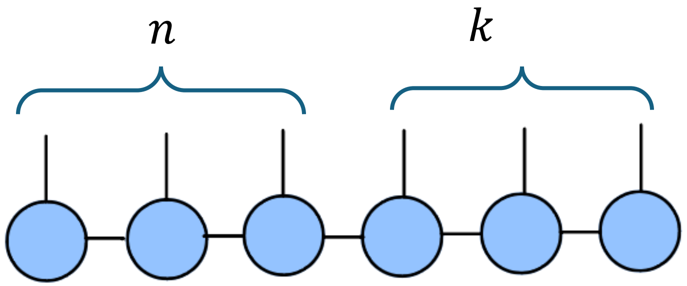
NN_QC/Tensor_train/codes/Pi_VS_E.png

$\beta = 100$

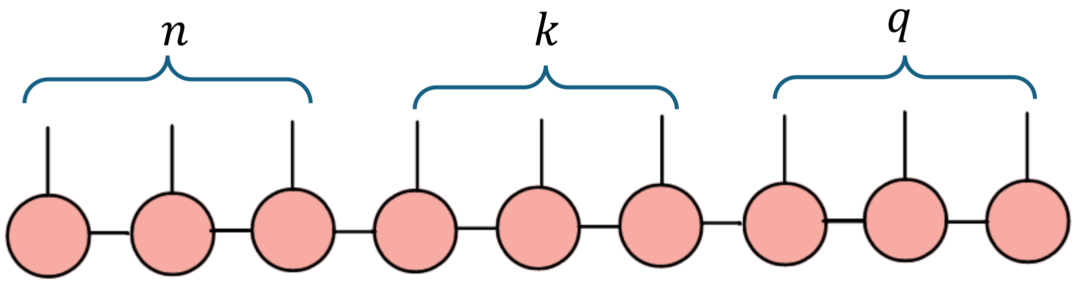
Can we do this more efficiently?

Zipup algorithm

$$G(i\omega_n, k) =$$



$$G(i\omega_n, k+q) =$$



$$\Pi(i\nu, q) = -\frac{\mathrm{T}}{2^{\mathrm{L}_2+1}} \sum_{k=-2^{L_2}}^{2^{L_2}} \sum_{n=-2^{L_1}}^{2^{L_1}} G(i\omega_n, k) G(i\omega_n + i\nu, k+q)$$

examples

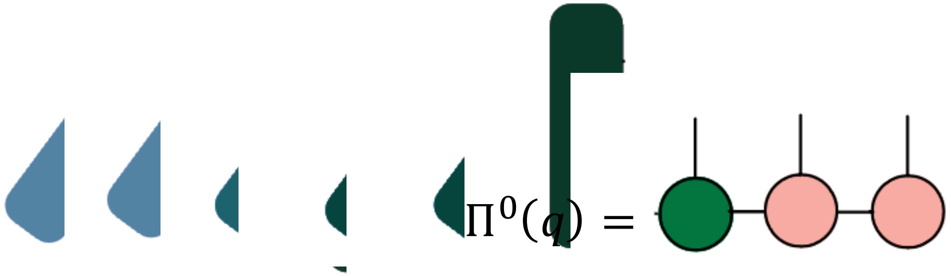
NN_QC/Tensor_train/TCI_polarization_zipup/TCI_polarization_1D_zipup.ipynb

$$\epsilon(p) = p^2 - \mu$$

NN_QC/Tensor_train/TCI_polarization_zipup/TCI_polarization_1D_zipup copy.ipynb

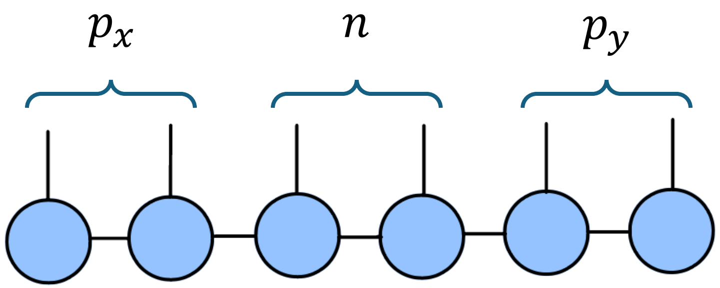
$$\epsilon(p) = \cos(p) - \mu$$

$$\Pi^0(q) =$$

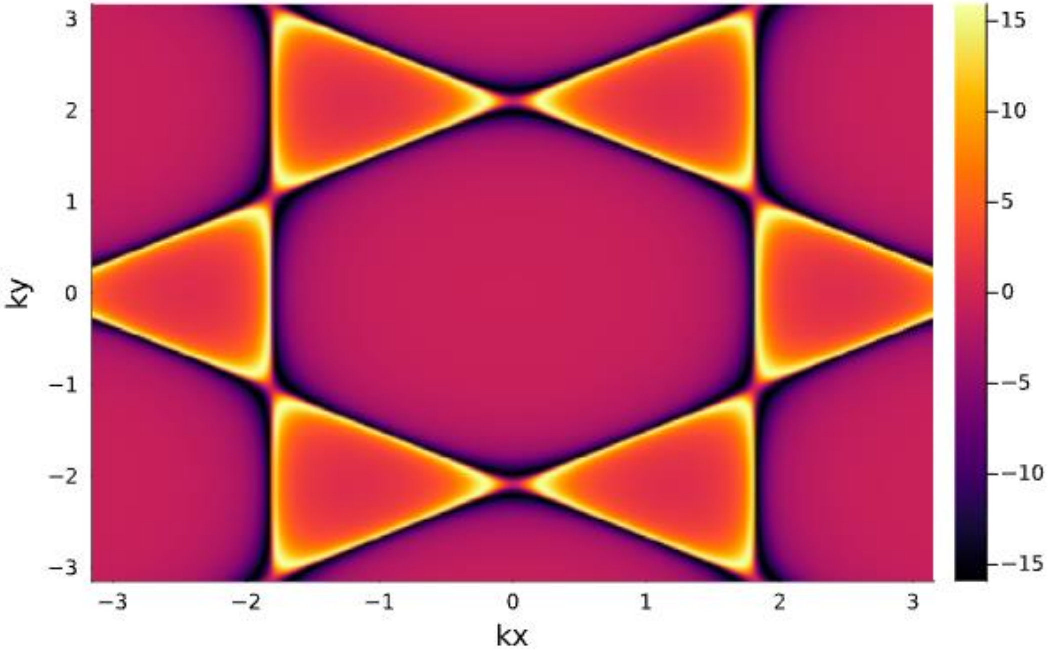


2D polarization with Graphene dispersion

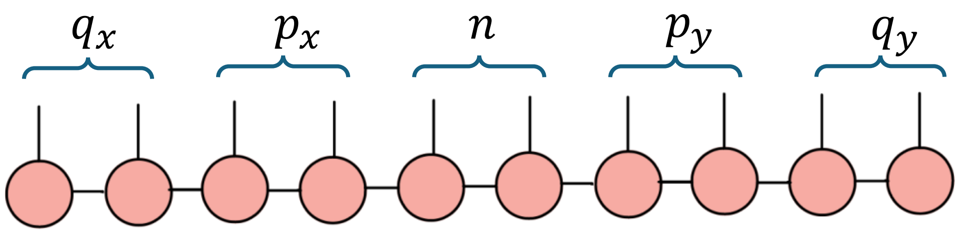
$$G^{\beta}(n, p_x, p_y) = \frac{1}{i\omega_n \; - \epsilon(p_x, p_y)}$$



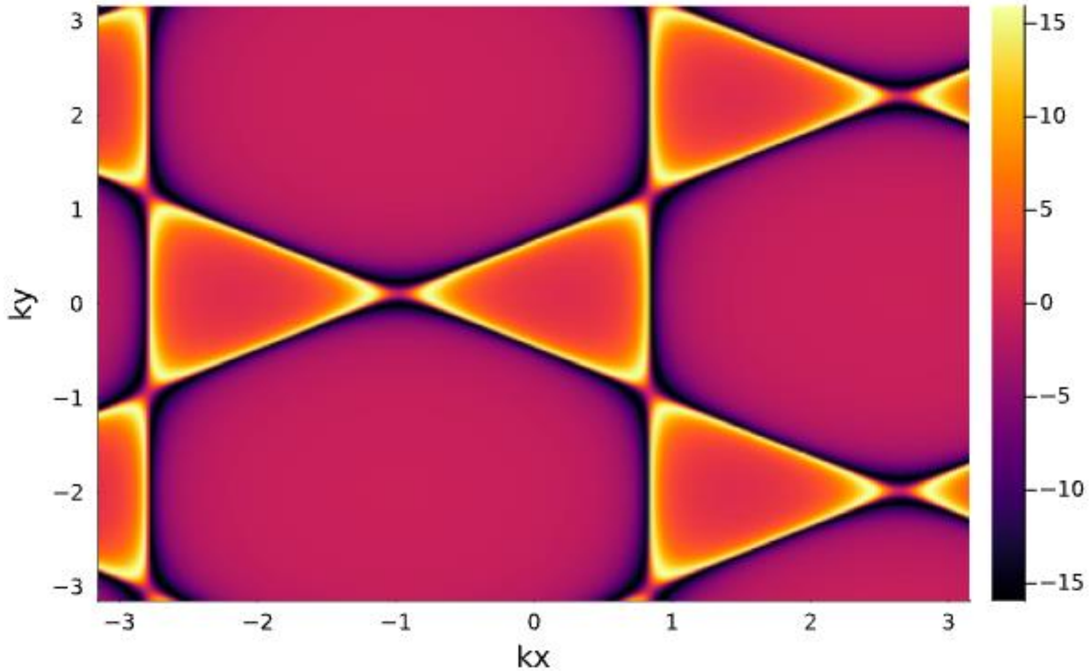
$G^{100}(0, p_x, p_y)$



$$G^{\beta}(n, p_x, p_y, q_x, q_y) = \frac{1}{i\omega_n \; - \epsilon(p_x + q_x, p_y + q_y)}$$



$G^{100}(0, p_x, p_y, 1, 2)$

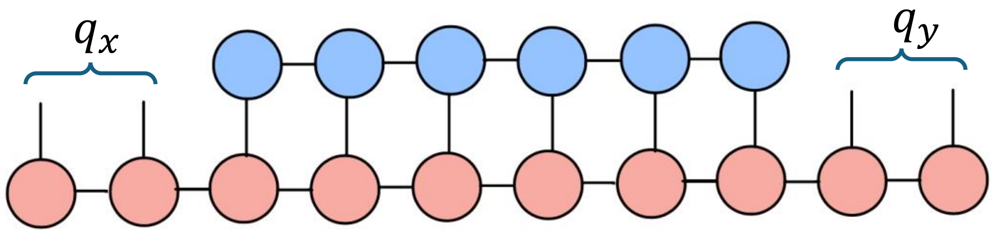


2D Polarization

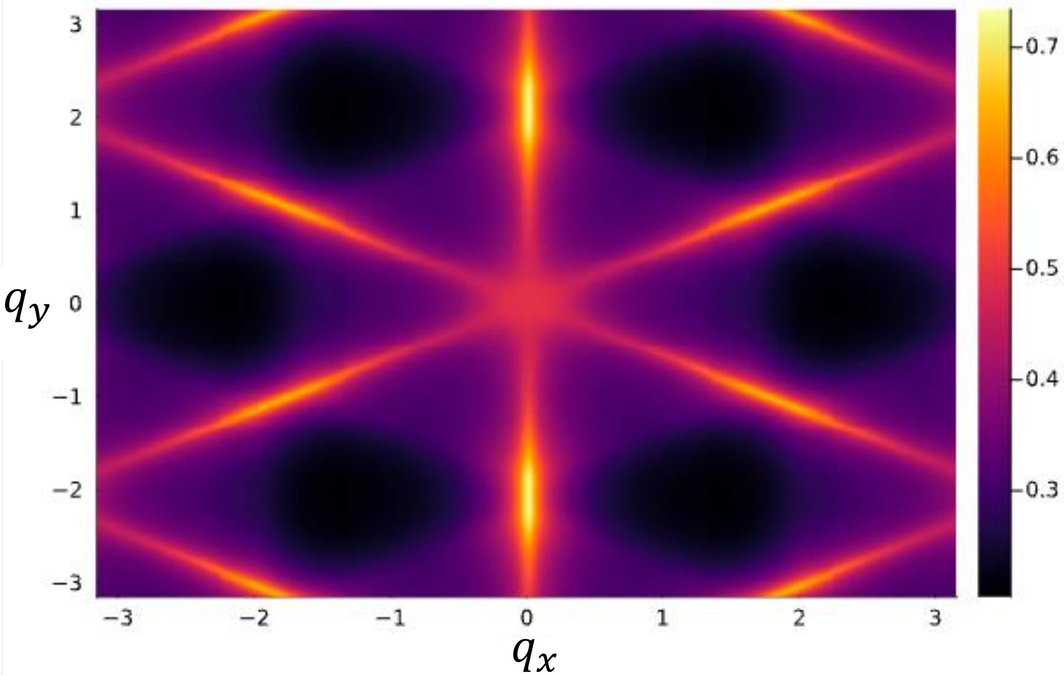
$$\Pi^0(\vec{q}) = -\frac{\mathrm{T}}{2^{L_{k_x}+L_{k_y}}} \sum_{p_x=-2^{L_{p_x}}}^{2^{L_{p_x}}} \sum_{p_y=-2^{L_{p_y}}}^{2^{L_{p_y}}} \sum_{n=-2^{L_n}}^{2^{L_n}} G\big(i\omega_n, p_x, p_y\big) G(i\omega_n, p_x+q_x, p_y+q_y)$$

$\beta = 100, L_{q_x} = 8, L_{p_x} = 10, L_n = 25, L_{p_y} = 10, L_{q_y} = 8$

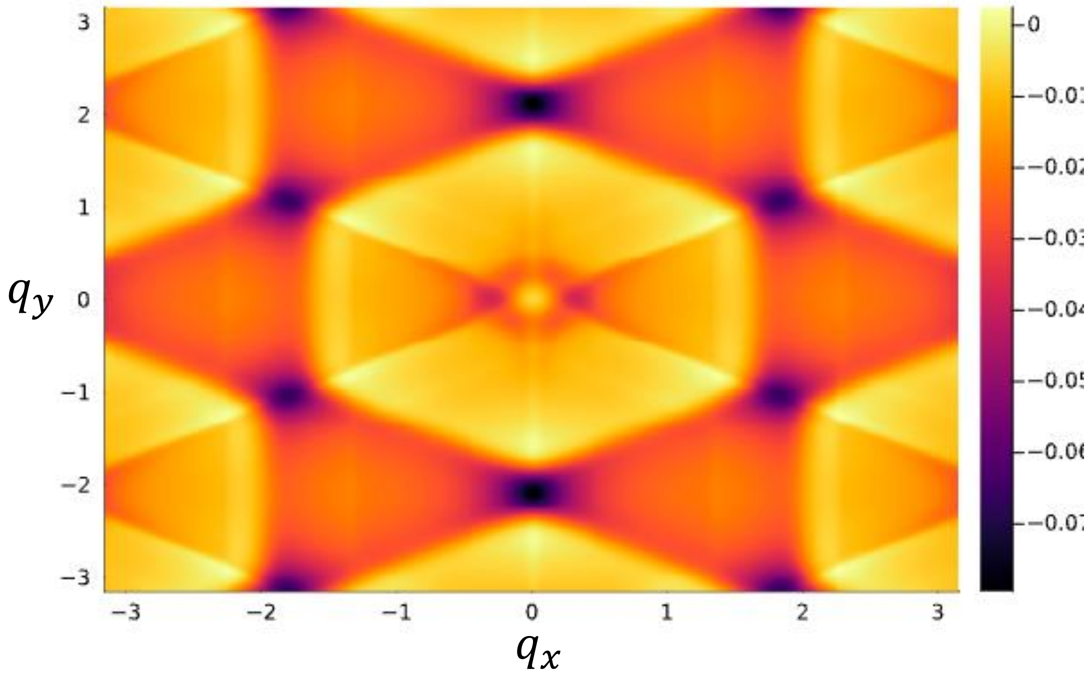
$\Pi^0(\vec{q}) =$



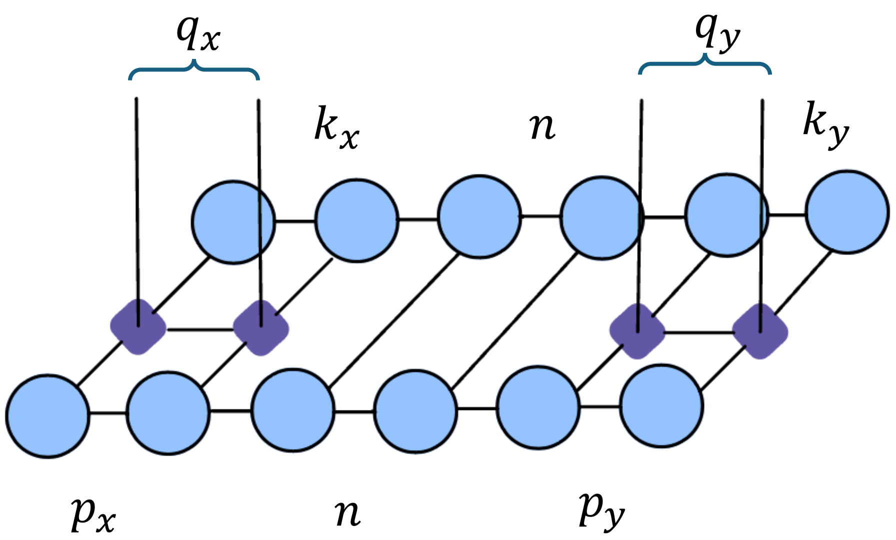
$Real(\Pi^0(\vec{q}))$

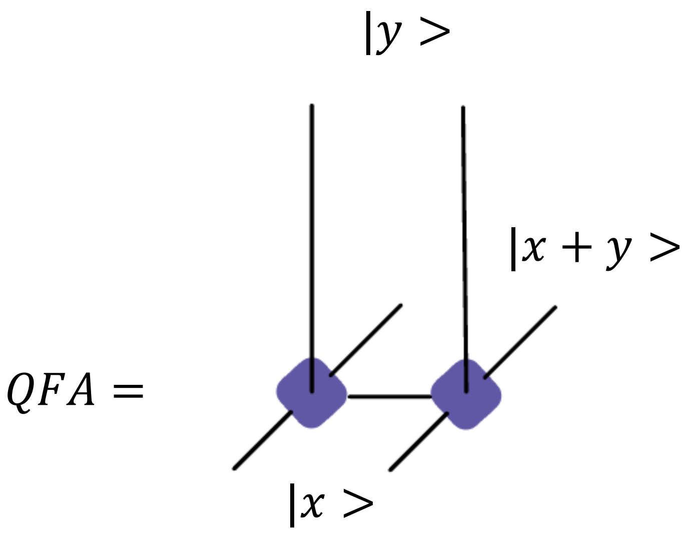


$imag(\Pi^0(\vec{q}))$



Can quantum-inspired algorithms be useful even without a quantum computer?





Quantum arithmetic with the Quantum Fourier Transform

Lidia Ruiz-Perez, Juan Carlos Garcia-Escartin

Low-rank tensor decompositions of quantum circuits

Patrick Gelß

$$\sum_{\boldsymbol{k}}\sum_{\boldsymbol{p},n} G(n,p_x,p_y)G(n,k_x,k_y)\underbrace{\delta(p_x+q_x-k_x)\delta(p_y+q_y-k_y)}_{QFA}$$

End