Tensor Networks for in medium particle dynamics

Diogo André **Marco Magalhães** **Pedro Oliveira**

Supervisor: Marco Leitão

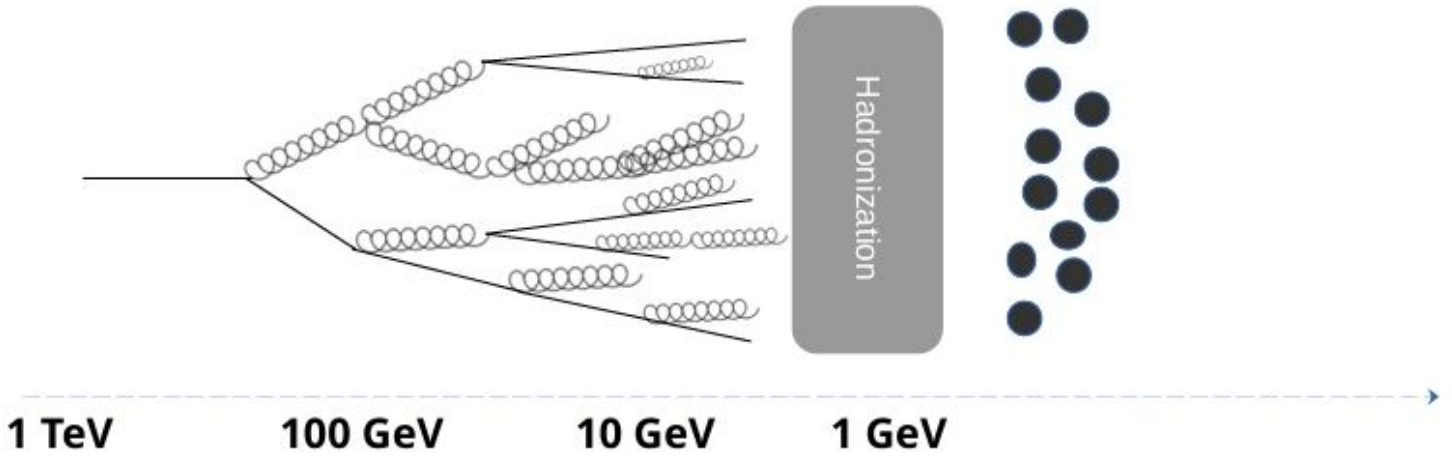
LIP – Laboratório de Instrumentação e Física Experimental de Partículas

Summer 2026

Quantum Chromodynamics (QCD) and Quark-Gluon Plasma (QGP)

QCD is the theory that predicts the behavior of **quarks** and **gluons** interacting via the **strong force**.

Depending on the energy scale, these particles can be found in **deconfined** state (**partons** – **high** energies), or in a **confined** arrangement (**hadrons** – **low** energies).

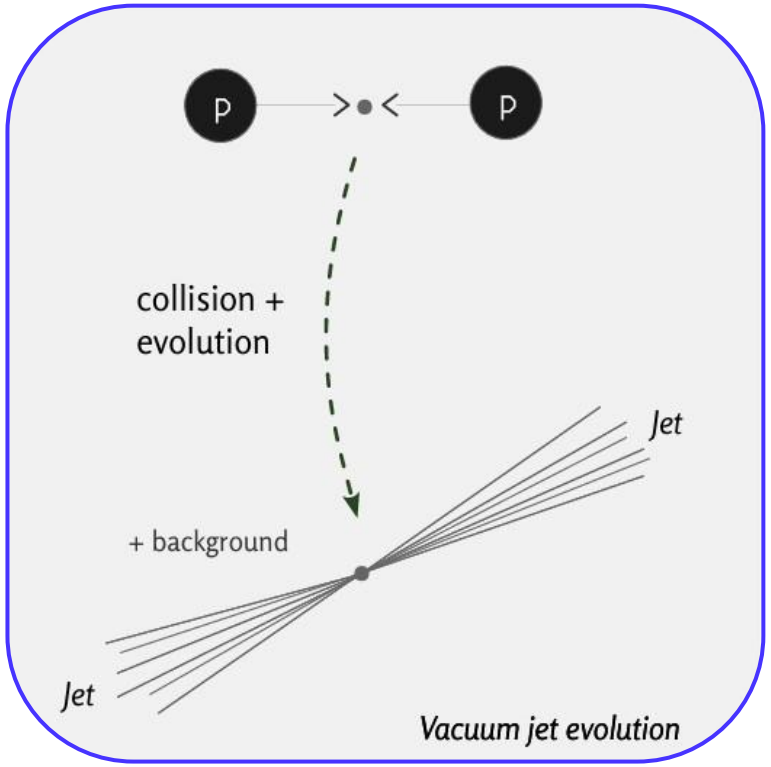


When at high energies, according to QCD, partons can radiate other partons of, progressively, lower energies until they reach the **hadronization scale**.

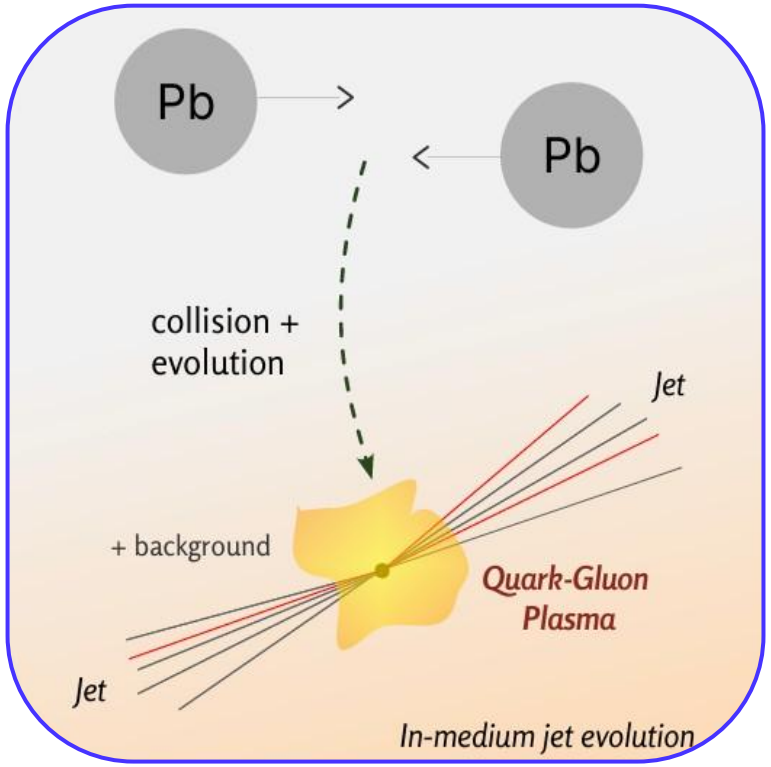
Quantum Chromodynamics (QCD) and Quark-Gluon Plasma (QGP)

At extreme values of temperature and density, partons can be “found” in a quasi-free state, the **Quark-Gluon Plasma**, also known as the “primordial liquid”, since it was the **universe’s first state of matter** for its first microseconds of existence.

Proton - Proton



Heavy-ion



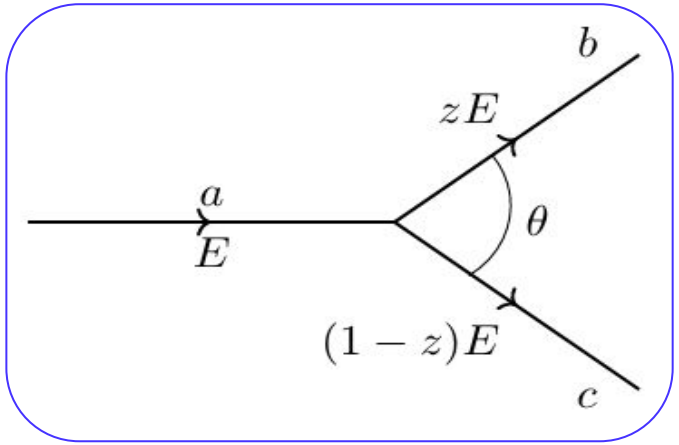
QGP can be recreated in **Heavy-Ion collisions!**

Quantum Chromodynamics (QCD) and Quark-Gluon Plasma (QGP)

Parton showers are intrinsically probabilistic, meaning that, each splitting is governed by a differential probability derived from QCD and dependent on the properties of the medium.

$$\mathrm{d}\mathcal{P}_{a\to bc} = \frac{\alpha_s}{2\pi}[zP_{a\to bc}(z)]\frac{\mathrm{d}z}{z}\frac{\mathrm{d}\theta^2}{\theta^2}$$

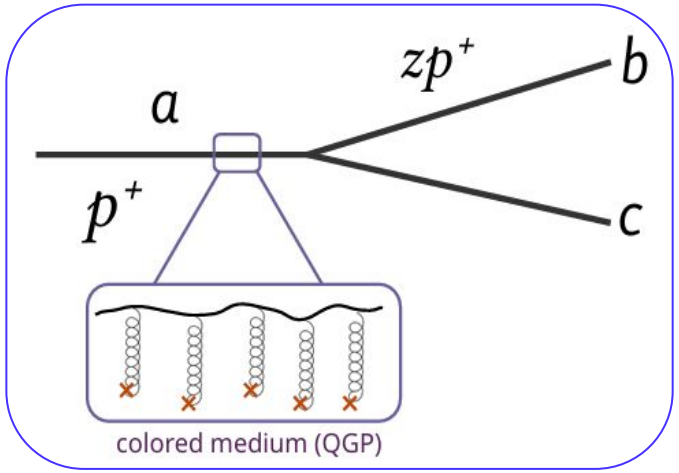
(vacuum)



The presence of a colored medium such as QGP leads us to modify the differential probability expression. In this work we used the following:

$$\mathrm{d}\mathcal{P}^{\text{in-medium}}_{a\to bc} = \mathrm{d}\mathcal{P}^{\text{vacuum}}_{a\to bc} \times \mathcal{R}_{\text{med}}(z,\theta)$$

$$\mathcal{R}_{med}(z,\theta) \sim \mathrm{Re}[\mathcal{B}(t,k,l=0)]$$



Quantum Chromodynamics (QCD) and Quark-Gluon Plasma (QGP)

In the previous slide, we saw that the differential probability depends on a B term. Such term's evolution is described by the following **4+1D** partial differential equation:

$$i\partial_t \mathcal{B}(t,\boldsymbol{k},\boldsymbol{l}) = \left[\frac{2\boldsymbol{k}\cdot\boldsymbol{l}}{\omega} - i\tilde{q}(\nabla^2_{\boldsymbol{k}} + \nabla^2_{\boldsymbol{l}})\right]\mathcal{B}(t,\boldsymbol{k},\boldsymbol{l}) + \mathcal{S}(t,\boldsymbol{k},\boldsymbol{l})$$

From the formalism of the problem, we can exploit the isotropy of the medium and reduce the equation's dimensionality to **3+1D**, expressing it in **polar coordinates**:

$$i\partial_t \mathcal{B}(t,k,l,\psi) = \left[\frac{2kl\cos\psi}{\omega} - i\tilde{q}\left(\frac{\partial^2}{\partial k^2} + \frac{1}{k}\frac{\partial}{\partial k} + \frac{\partial^2}{\partial l^2} + \frac{1}{l}\frac{\partial}{\partial l} + \left(\frac{1}{k^2}+\frac{1}{l^2}\right)\frac{\partial^2}{\partial \psi^2}\right)\right]\mathcal{B}(t,k,l,\psi) + \mathcal{S}(t,k,l,\psi)$$

$$\mathcal{R}_{med}(z,\theta) \sim \mathrm{Re}[\mathcal{B}(t,k=\omega\theta,l=\mathbf{0})]$$

Even though such simplifications help, when considering a model with a more realistic medium and more than one splitting, dimensions increase drastically and an inevitable **crisis of dimensionality** arises.

Crisis of dimensionality

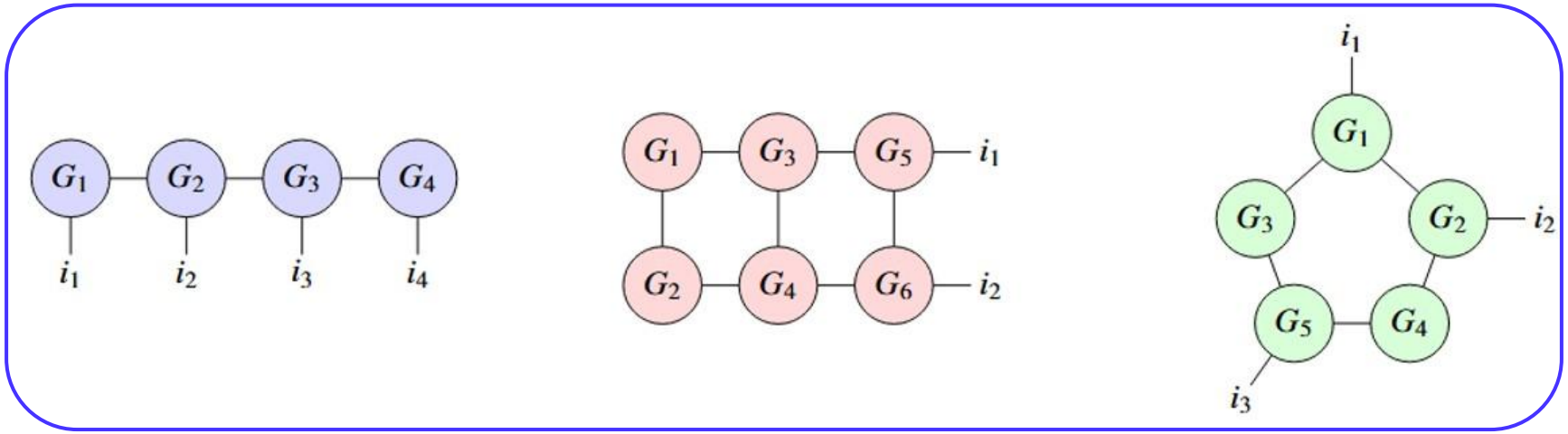
To understand the magnitude of this issue, let us consider a set of real numbers with N elements. In order to store this array, let's use a float32 format (4 bytes), and observe the memory requirements needed as we increase the rank of the set. For materialization purposes, let's say N = 5.000:

- r = 1: Mem. = 4*N = 20 kB → Fits in regular L1 CPU Cache (128 kB)
- r = 2: Mem. = $4*N^2$ = 100 MB → Fits in regular RAM (16GB)
- r = 3: Mem. = $4*N^3$ = 500 GB → Nearly overloads a regular SSD (512GB)
- r = 4: Mem. = $4*N^4$ = 2.5 PB → Much larger than the storage capacity of a regular PC
- …
- r = 23: Mem. > 10^{83} B → More bytes than the number of atoms in the universe (10^{72} up to 10^{83})

The memory capacity needed scales exponentially to the rank of the array, eventually leading to numbers that greatly surpass current and accessible storage hardware. To combat this issue, memory optimization algorithms are implemented, such as **Tensor Networks**, the ones approached in this work.

Tensor Networks and MPS/TT

Tensor Networks are factorizations of larger tensors into graph-like networks of smaller tensors. To visualize the different shapes of such arrangements, Tensor Diagram Notation was developed. Bellow, we provide some examples of this notation.



- G_1 Nodes / shapes: tensors; - G_1 — G_1 Connections: contractions; - i Free lines: tensor indices.

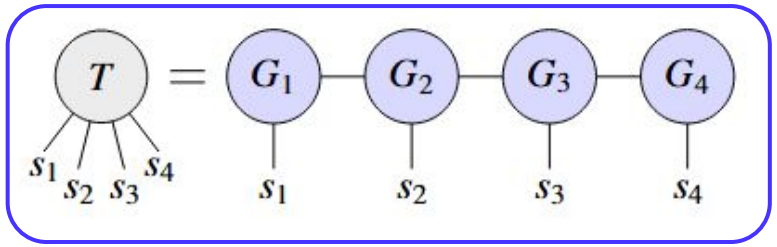
These can be organized in different ways, but in this internship, we make use of a kind of tree tensor network named **Matrix Product State**, or, due to its shape, **Tensor Train**.

Tensor Trains

In regards to properties, TT's are tree tensor networks, so, by definition, they produce only **one path between cores**, making it impossible for the presence of loops/cycles like the ones present in the two exemples before.

$$T^{s_1 s_2 s_3 s_4} = \sum_{\{\alpha\}} G^{s_1}_{\alpha 1} G^{s_2}_{\alpha_1 \alpha_2} G^{s_3}_{\alpha_2 \alpha_3} G^{s_4}_{\alpha_3}$$

$\leftrightarrow$



By far, the best contribution from the TT factorization comes from the memory cost. In its full form, the previous tensor, of arbitrary index dimension d, has a scaling factor of $\mathbf{O(d^N)}$, however, its TT version's cost scales according to $\mathbf{O(Ndm^2)}$, where m is the arbitrary bond dimension (or TT-rank) between cores.

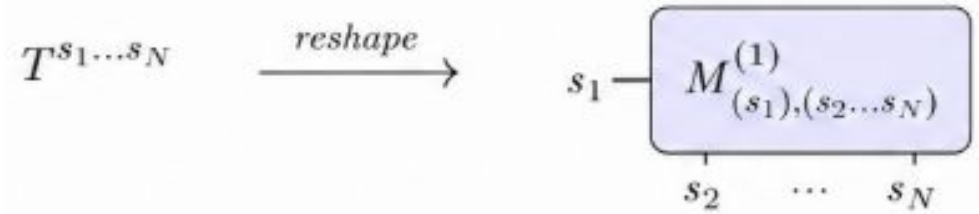
The TT shape introduces another optimization method for memory optimization: **truncation of elements** whose contributions to the final tensor have little weight according to a certain **tolerance input**. This "selection" of most valuable elements comes from the **TT-SVD algorithm** from which tensors are converted to TT's.

TT-SVD Algorithm

To better understand the mechanism that leads to the factorization of a tensor with N indices, let us be guided by the following steps, with corresponding diagrams:

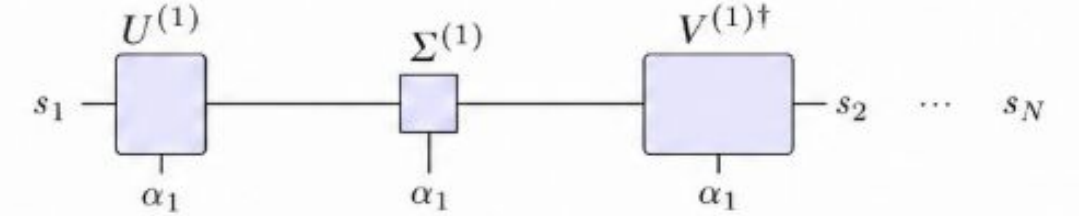
Starting with the original tensor, we reshape it grouping the first index as rows and the rest as columns.

$$T^{s_1 \ldots s_N} \xrightarrow{reshape} M^{(1)}_{(s_1),(s_2 \ldots s_N)}$$



To this matrix we apply **thin singular value decomposition** (SVD) and end up with three distinct matrices:

$$M^{(1)}_{(s_1),(s_2 \ldots s_N)} = U^{(1)} \Sigma^{(1)} V^{(1)\dagger}$$



From this decomposition, we truncate each matrix according to the contribution of each singular value from de diagonal matrix, remaining *r* singular values. Globally, the singular values eliminated should satisfy:

$$\sum_i \sigma_i^2 \leq \epsilon^2$$

ϵ is the absolute tolerance

As for each step:

$$\sum_{i \geq r} \sigma_i^2 \leq \delta^2 \qquad \delta = \epsilon / \sqrt{N-1}$$

δ is the tolerance for each step

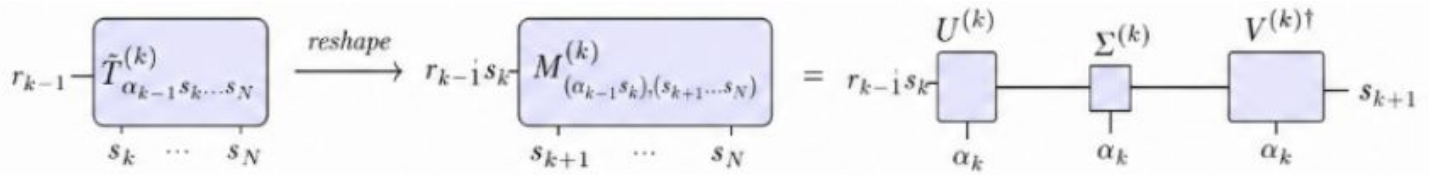
TT-SVD Algorithm

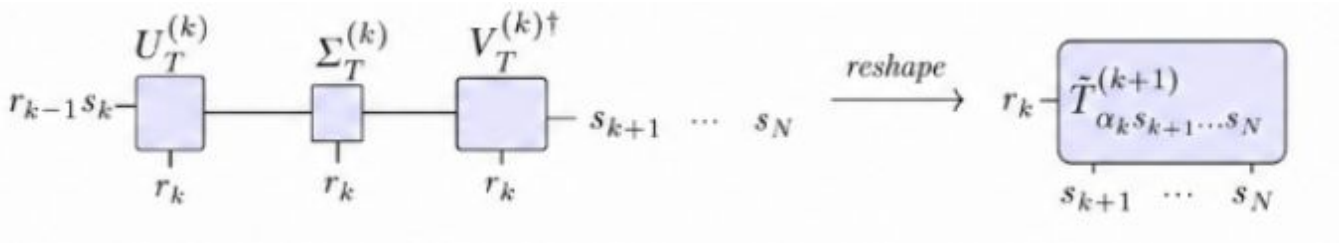
Post SVD and truncation, we extract the first core from the U_T matrix, and the following tensor becomes the reshaped tensor resultant from the product of $\sum_T$ with V_T. The recursive cycle after the first step can be expressed in the steps:

$$\tilde{T}^{\alpha_{k-1} s_k \ldots s_N} \xrightarrow{reshape} M^{(k)}_{(\alpha_{k-1} s_k),(s_{k+1} \ldots s_N)} = U^{(k)} \Sigma^{(k)} V^{(k)\dagger}$$

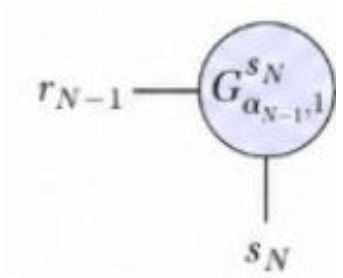
$$U_T^{(k)} \xrightarrow{reshape} G^{s_k}_{\alpha_{k-1},\alpha_k}$$

$$\Sigma_T^{(k)} V_T^{(k)\dagger} \xrightarrow{reshape} \tilde{T}^{(k+1)}_{\alpha_k s_{k+1} \ldots s_N}$$





Until we compute core N-1. For the last core, there's no need for SVD or further reshaping, since we reach a matrix with adequate shape, $G^{s_N}_{\alpha_{N-1}}$.



The global error is given by the Frobenius norm: $error = \|T - TT\|_F \leq \sum_{discarted} \sigma_i^2$

Study of the Memory Requirements of the Source Term

To start our implementation, we start with an example of the introduced algorithm applied to our equation's source term. This corresponds to the harmonic-oscillator approximation that, in polar coordinates, looks like:

$$\mathcal{S}(t,k,l,\psi) = 2\omega \frac{k^2 - l^2}{k^2 + l^2 + 2kl\cos\psi} \left(1 - e^{-\frac{i}{2\omega\Omega}\tan(\Omega t)\,(k^2+l^2+2kl\cos\psi)}\right)$$

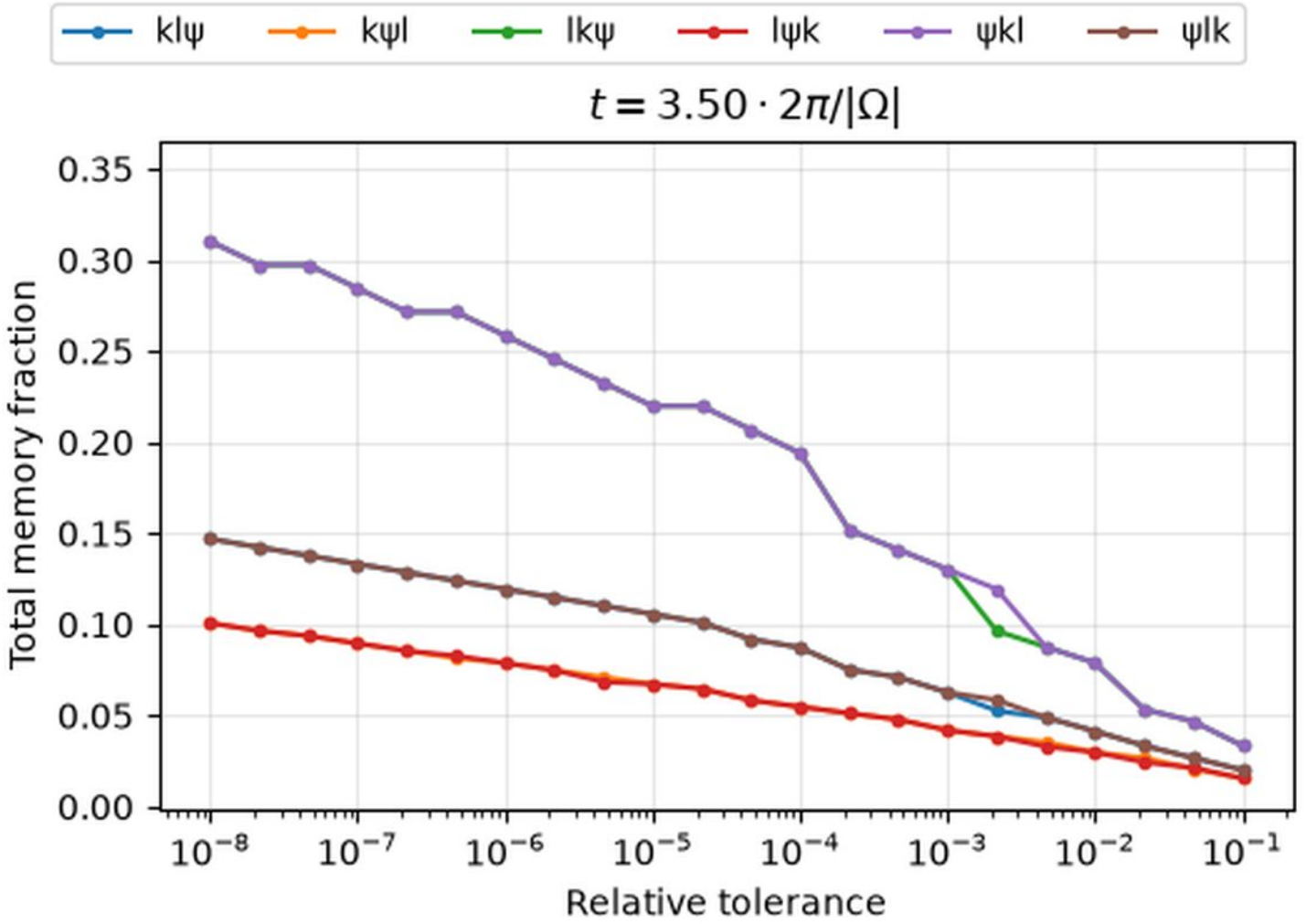
where $\Omega = (1-i)/2\sqrt{\tilde{q}/\omega}$.

The first natural step is to analyse how factorizable this tensor is, for the best TT application in our equation's evolution.

Study of the Memory Requirements of the Source Term

- Compression is more or less independent of time
- After discretization, index permutations where ψ is in the middle achieve the best results
- TMF < 10% for RTs ~ 10^{-6}

$$\|S\|_F \approx 30000$$



The need for the MPO

Depending on the propagator, the evolution of an equation may not be fully factorizable via TT. This point and the reason behind it becomes very clear once we compare the discretizations of a perfectly separable (heat equation) and a non-separable propagator (internship equation).

	Heat	B-term
Analytical expression	$\alpha\left(\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2}+\frac{\partial^2}{\partial y^2}\right)u$	$\left[\frac{2kl\cos\psi}{\omega}-i\tilde{q}\left(\frac{\partial^2}{\partial k^2}+\frac{1}{k}\frac{\partial}{\partial k}+\frac{\partial^2}{\partial l^2}+\frac{1}{l}\frac{\partial}{\partial l}+\left(\frac{1}{k^2}+\frac{1}{l^2}\right)\frac{\partial^2}{\partial\psi^2}\right)\right]F$
Discrete expression	$D_{ijk,i'j'k'}u_{i'j'k'}=\left\{D^{(2)}_{ii'}\delta_{jj'}\delta_{kk'}\right.$ $=\delta_{ii'}D^{(2)}_{jj'}\delta_{kk'}$ $\left.=\delta_{ii'}\delta_{jj'}D^{(2)}_{kk'}\right\}u_{i'j'k'}$	$(H_hF)_{ijm}=\frac{2k_il_j\cos\psi_m}{\omega}F_{ijm}$ $-\mathrm{i}\tilde{q}\left[\sum_p(R_k)_{ip}F_{pjm}+\sum_q(R_l)_{jq}F_{iqm}\right.$ $\left.+\left(k_i^{-2}+l_j^{-2}\right)\sum_s(D_\psi)_{ms}F_{ijs}\right]$

As you can observe, the B-term (F for application purposes) propagator has mixed indices in its discretization, not allowing for the total decomposition of the solution, therefore, we must implement a way of evolving a multiple core factorization, the MPO.

MPO formulation

We start by **reshaping** the H operator in to T, with **grouped input and output indices**. After applying TT-SVD, we reach the core shape, to finally unflatten said cores, by the grouped indices and achieving the final step of the factorization. As observed in the following diagram:

$$H^{i_1 \dots i_d}_{j_1 \dots j_d} \xrightarrow{\text{reshape}} T^{(i_1,j_1)\dots(i_d,j_d)} \downarrow TT\text{-}SVD \downarrow G_H[n]^{k_{n-1} i_n}_{k_n j_n} \xleftarrow{\text{reshape}} G_T[n]^{k_{n-1}(i_n,j_n))}_{k_n}$$

Now it is possible to express the application of H on to a state F, as the same application of the H cores to the F cores:

$$G_{HF}[n]^{(k_{n-1},l_{n-1})i_n}_{(k_n,l_n)} = G_H[n]^{k_{n-1} i_n}_{k_n j_n} G_F[n]^{l_{n-1} j_n}_{l_n}$$

MPO formulation

Since the sum of TT factorizations is made up of multiple cores, we can establish the analogy to the sum of MPO operator in the expression:

$$(S+T)^{i_1 \dots i_d} = \begin{pmatrix} S[1] & T[1] \end{pmatrix}^{i_1} \begin{pmatrix} S[2] & 0 \\ 0 & T[2] \end{pmatrix}^{i_2} \cdots \begin{pmatrix} S[d-1] & 0 \\ 0 & T[d-1] \end{pmatrix}^{i_{d-1}} \begin{pmatrix} S[d] \\ T[d] \end{pmatrix}^{i_d}$$

This way, the sum is arranged by blocks and matrix products, yielding the core product of S and T.

Something worth noting in the MPO application is that each individual core computation **scales the dimension of the connecting space.** To correct this intermediate scaling, we apply a **truncation** process similar to the one present in the TT-SVD algorithm, effectively optimizing for the amount of data that requires storage.

Numerical Integration of The B-term Equation

We must, then, build a solution suitable to the incorporation of the MPO, starting by noting that the equation has a Schrodinger-like form, with a non-hermitian time independent Hamiltonian.

$$i\partial_t \boldsymbol{B}(t) = \hat{H}\boldsymbol{B}(t) + \boldsymbol{S}(t),$$

The recursive solution to such equations has the form:

$$\boldsymbol{B}(t+\delta t) = e^{-i\hat{H}t}\boldsymbol{B}(t) - i\int_t^{t+\delta t} e^{-i\hat{H}(t+\delta t-t')}\boldsymbol{S}(t')dt'.$$

Finally, applying Simpson's rule of integration yields:

$$\boldsymbol{B}(t+\delta t) = e^{-i\hat{H}\delta t}[\boldsymbol{B}(t) - \frac{i}{6}\boldsymbol{S}(t)\delta t] - \frac{4i}{6}e^{-i\hat{H}\delta t/2}\boldsymbol{S}(t+\delta t/2)\delta t - \frac{i}{6}\boldsymbol{S}(t+\delta t)\delta t + \mathcal{O}(\delta t^5).$$

Therefore we need to compute terms of the form $e^{-i\hat{H}\delta\tilde{t}}\boldsymbol{F}$ and that's where the MPO will be applied.

Faber turns propagation into repeated H-actions

For the non-Hermitian Hamiltonian, Faber applies the exponential to a TT vector through a polynomial expansion; the matrix exponential itself is never formed.

$$A = \frac{\widehat{H}}{\rho} - \gamma_0 I$$

Approximate the exponential action — not the exponential:

$$e^{-ih\widehat{H}} F \approx \sum_{n=0}^{N} c_n(h)\, \Phi_n$$

Generate the Faber basis recursively in TT form:

$$\begin{aligned} \Phi_0 &= F, & \Phi_1 &= A\Phi_0, \\ \Phi_2 &= A\Phi_1 - 2\gamma_1\Phi_0, \\ \Phi_{n+1} &= A\Phi_n - \gamma_1\Phi_{n-1}, & n &\geq 2. \end{aligned}$$

One MPO–TT action per polynomial order; no dense Hamiltonian or matrix exponential is formed. $\widehat{H}\Phi_n$

The Hamiltonian as a sum of simple MPOs

This is the important case for our solver: the reduced operator is a short exact sum, not one separable product. Its five terms are encoded once as a block MPO.

	Rank-1 terms	Block MPO
Operator structure	$\widehat{H} = \sum_{\mu=1}^{5} W_\mu, \qquad \mathrm{rank}_{\mathrm{MPO}}(W_\mu) = 1$	$\widehat{H} = \frac{2}{\omega} K \otimes L \otimes C - i\tilde{q}\big[R_k \otimes I_\ell \otimes I_\psi + I_k \otimes R_\ell \otimes I_\psi + P_k \otimes I_\ell \otimes D_\psi + I_k \otimes P_\ell \otimes D_\psi\big]$
Action on a TT	$\sum_{u=1}^{5} W_\mu \implies W, \qquad \mathrm{rank}_{\mathrm{MPO}}(W) = (4,3)$	$\widehat{H} F^{\mathrm{TT}} = W F^{\mathrm{TT}}, \qquad (r_1, r_2) \xrightarrow{\widehat{H}} (4r_1, 3r_2)$

Encode the five terms once. Every Faber step contracts the block MPO with the TT, forms the recurrence sum, and rounds only the completed result.

One step is validated; the full solver is next

Faber is one component of the project. The target is a controlled evolution from the compressed source to the physical observable.

$$S(t) \xrightarrow{\text{TT–SVD}} S^{\text{TT}}(t) \xrightarrow{\text{MPO+Faber}} U_h S^{\text{TT}}(t) \xrightarrow{\text{Simpson}} B(t+h) \xrightarrow{\text{iterate}} B(L) \longrightarrow R_{\text{med}}$$

1. Validated step: 10×8×6, h=0.050 fm/c, N=31, relative error 9.38×10^{-11}.

2. Next: use MPO–Faber at Simpson nodes; march to L and test time/grid convergence.

3. Then: validate polar-origin recovery and accumulated error before extracting R□ed.

TT source → MPO–Faber → R□ed

Conclusion: a tensor-network route to the observable

The project has established the compressed representations and the one-step propagation kernel; the remaining task is end-to-end evolution.

VALIDATED

Source in TT

TT-SVD compresses the source with a controlled approximation error.

→

EXACT

Hamiltonian MPO

Five rank-one terms form one block MPO with bond ranks (4, 3).

→

VALIDATED

Faber step

Exponential action → repeated MPO–TT products; one step reaches 9.38×10^{-11}.

→

NEXT

Full observable

March to L and validate the polar-origin recovery of R□ed.

Take-home message

Dense operators are replaced by controlled TT/MPO operations, leaving a clear path from the source term to the physical observable.

Tensor Networks for in medium particle dynamics

Thank you for your attention

Let there be questions!!

References

- M. Leitão, J. G. Milhano, and A. Soto-Ontoso, "In-medium QCD splittings beyond the soft, large-Ncand harmonic-oscillator approximations all at once". In (June 2026) arXiv: 2606.26039 [hep-ph].
- Unknwon, Welcome to the Tensor Network, https://tensornetwork.org/ (2026), accessed: 18-8-2026

Extra resources

TT-SVD condensed diagram explanation: