Quantized Tensor Trains for Integral Equations

Michael Kielstra

$$a\sigma(x) + \int_{\Omega} K(x,y)\sigma(y)d\Omega_y = f(x)$$

$$A\sigma = f$$

Uses:

- Various physical phenomena (especially $K(x,y) = \kappa(|x-y|)$).
- *Mesh-free methods* for PDEs.

Quick Reminder: Solving Matrix Equations

GMRES, CG, *et al.* minimize a residual norm $\sigma \mapsto ||A\sigma - f||$.

To compute this, we only need a map $\sigma \mapsto A\sigma$.

If A is poorly-conditioned, this can converge slowly.

If we can define P such that $P(x) \approx A^{-1}x$, we can solve $P(A)\sigma = P(f)$ instead.

The Plan

Idea: Use a QTT to represent A.

1. Factor A into a QTT.
2. Do matvecs by QTTs (which lets us invert A with GMRES).
3. Do approximate inverses in QTT form (which lets us precondition GMRES).

Step 1: Quantizing A

Idea: Split A into a grid which defines a *source tree* and a *target tree*.

Definition: A box in the source tree and a box in the target tree are *well-separated* if the distance between them is at least the diameter of the largest box.

Intuition: Well-separated source-leaf/target-leaf pairs are far from the main diagonal of the matrix A.

Definition: A is *FMM-compressible* if any matrix sub-block representing a well-separated pair of boxes has maximum numerical rank k_ϵ for some fixed k_ϵ.

Tensorizing A

Form the *product tree*, where every node represents a node in the source and the target trees at once.

Think of A as a tensor where each element is given by a path through the product tree.

Note that unfolding matrices of A are not the same as the original matrix!

Does This Work?

Definition: A kernel $K(x, y)$ is *translation-invariant* if $K(x + \omega, y + \omega) = K(x, y)$ for all ω.

Theorem: Let A be FMM-compressible and let the kernel K be translation-invariant. Then the QTT representation has bounded rank $r = \max(r_k) \leq k_\epsilon^2 + 4D - 1$, where D is the dimension of the ambient space.

Step 2: Matvecs by QTT-Compressed Matrices

Let A have cores given by $G_k^A(\alpha_{k-1}, \overline{i_k j_k}, \alpha_k)$ and b have cores given by $G_k^b(\beta_{k-1}, j_k, \beta_k)$. Then Ab has cores given by

$$G_k^{Ab}(\overline{\alpha_{k-1}\beta_{k-1}}, i_k, \overline{\alpha_k\beta_k}) = \sum_{j_k} G_k^A(\alpha_{k-1}\overline{i_k j_k}\alpha_k) G_k^b(\beta_{k-1} j_k \beta_k).$$

$O(r_A^2 r_b^2 N \log N)$.

Well, that was easy.

Step 2, Shia Surprise: Matvecs with Uncompressed Vectors

Idea: Split b up along the target tree and use QTT cores to move it to the source tree.

Initialize: $y_0 = b^T = y_0(\alpha_0, \overline{j_1 j_2 \cdots j_d})$

Core iteration:

1. $M_k(\overline{\alpha_k i_k}, \overline{\alpha_{k-1} j_k}) = G_k^A(\alpha_{k-1}, \overline{i_k j_k}, \alpha_k)$
2. $b_k(\overline{\alpha_{k-1} j_k}, \overline{i_1 \cdots i_{k-1} j_{k+1} \cdots j_d}) = y_{k-1}(\alpha_{k-1}, \overline{i_1 \cdots i_{k-1} j_k \cdots j_d})$
3. $\phi_k = M_k b_k$
4. $y_k(\alpha_k, \overline{i_1 \cdots i_k j_{k+1} \cdots j_d}) = \phi_k(\overline{\alpha_k i_k}, \overline{i_1 \cdots i_{k-1} j_{k+1} \cdots j_d})$

$O(r_A^2 N \log N)$.

Step 3: (Approximate) Inversion of QTT-Compressed Matrices

Idea: Consider $AX = I$ and solve for X.

Useful identity: $vec(ABC) = (C^T \otimes A)vec(B) \implies (I \otimes A)vec(X) = vec(I)$

Write X as a QTT, fix all cores but one, and solve for that core. Repeat for all cores.

Problem: can't change the ranks of X.

One possible solution (DMRG): solve for two cores at once, then split them up.

Reference

Corona, E., Rahimian, A., & Zorin, D. (2017). A Tensor-Train accelerated solver for integral equations in complex geometries. *Journal of Computational Physics*, 334, 145–169. https://doi.org/10.1016/j.jcp.2016.12.051