Fast BEM solution for scattering problems using Quantized Tensor Train format

Jean-René Poirier, Ayoub Bellouch, Olivier Coulaud, Oguz Kaya

To cite this version:
Jean-René Poirier, Ayoub Bellouch, Olivier Coulaud, Oguz Kaya. Fast BEM solution for scattering problems using Quantized Tensor Train format. COMPUMAG 2019 - 22nd International Conference on the Computation of Electromagnetic Field, Jul 2019, Paris, France. hal-02264277

HAL Id: hal-02264277
https://hal.archives-ouvertes.fr/hal-02264277
Submitted on 19 Sep 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Fast BEM solution for scattering problems using Quantized Tensor-Train format

J.-R. Poirier¹,², A. Bellouch², O. Coulaud², and O. Kaya³
¹LAPLACE, Université de Toulouse, CNRS, INPT, UPS, Toulouse, France
²INRIA Bordeaux Sud-Ouest-HIEPACS
³Université Paris Saclay

It is common to accelerate the boundary element method by compression methods (FMM, H-Matrix / ACA) that enable a more accurate solution or a solution in higher frequency. In this work, we present a compression method based on a transformation of the linear system into Tensor-Train format by the quantization technique. The method is applied to a scattering problem by a canonical object with a regular mesh and improves the performance obtained by the previous methods.

Index Terms—Boundary integral equations, electromagnetic diffraction, Quantized Tensor-Train

I. INTRODUCTION

MANY applications in science and engineering are formulated in terms of boundary integral equations. This is namely the case in electromagnetic scattering to avoid the use of an artificial condition to truncate the domain of study. From a numerical point of view, the main difficulty is the solution of a full complex linear system

$$Ax = b.$$  

Many compression and solution accelerations of this linear system have been developed and allowed to increase the maximum accessible sizes from a few thousand to several millions. Continuing to improve performance and accelerate solution remains a topical issue. In this work we propose the use of a tensor format for solving the algebraic problem.

As numerical example, we will consider the scattering by a perfectly conducting cylinder \( \Gamma \) in the E-polarization \( (u = E_z) \). The involved boundary-value problem to solve is then the Helmholtz equation with a Dirichlet boundary condition and a radiation condition at infinity.

The BEM solution can be written with a single layer potential [1],

$$\int_\Gamma G(x, x_s) j(x_s) \, dy(x_s) = -u^{inc}(x), \forall x \text{ on } \Gamma,$$  

where \( \Gamma \) is the boundary, \( u^{inc} \) the incident electric field, \( j \) the sought density current and \( G \) the Green’s function.

In free space, this Green function is usually given by \( G(x, x_s) = \frac{1}{4\pi} H_0^2(k|x - x_s|) \) where \( i \) is the imaginary unit, \( k \) the wave number and \( H_0^2 \) the Hankel function of second kind.

II. SOLUTION WITH TENSOR TECHNIQUES

For many years various types of problems have been formulated using tensors instead of a classical linear matrix algebra [3]. More recently, tools have been developed to treat high order tensors (dimension higher than 3) based on the implementation of low rank techniques to effectively reproduce the algebraic structure of the system. In the literature, there are two main formats in terms of tensors for an efficient and stable “hollow” representation of a very large system. These are the approximations by Tensor-Train (TT) and Hierarchical Tucker decompositions. In this work we have chosen to apply the TT format to formulate the integral equation problem for a fast solution.

A. Tensor-Train Format

The TT format [2] consists of writing a \( d \)-dimensional tensor \( T \in \mathbb{R}^{d_1 \times \cdots \times d_n} \) as a chain of 3-dimensional tensors according to the formula

$$T(i_1, \ldots, i_d) = \sum_{a_0, \ldots, a_d} G_1(a_0, i_1, a_1) G_2(a_1, i_2, a_2) \cdots G_{d-1}(a_{d-1}, i_d, a_d)$$

which can be written in a compact form

$$T(i_1, \ldots, i_d) = \sum_{a_0, \ldots, a_d} G_1[i_1] G_2[i_2] \cdots G_{d-1}[i_{d-1}] r_0 \times r_1 r_1 \times r_2 \cdots r_{d-1} \times r_d$$

where

- \( G_i \): TT-cores (matrices of size \( r_{k-1} \times r_k \) with \( r_0 = r_d = 1 \))
- \( r_i \): TT - rank
- \( r = \max(r_i), r \) maximal rank of cores.

The complexity of the storage is \( O(d n r^2) \) for \( O(n^d) \) elements. If \( r \) is small the tensor is of low rank and the storage and as a result the solver will be fast.

The same format is given for a linear operator in \( \mathbb{R}^{d_1 \times \cdots \times d_n} \) which is represented by a \( 2d \)-dimensional tensor \( \mathcal{A} \) that couple elements as \((i_a, j_n)\) for \( n = 1 \ldots d \) resulting from the couple \((i_a, j_n)\). TT representation of this tensor has the compact form

$$\mathcal{A}(i_1, \ldots, i_d, j_1, \ldots, j_d) = M_1[i_1, j_1] M_2[i_2, j_2] \cdots M_d[i_d, j_d]$$

where \( M_k[i_k, j_k] \) is a matrix of rank \( r_{k-1} \times r_k \).

In TT format, we have the corresponding algebraic operations (addition, matrix-vector product, dot product, etc.) that are fast because of the reduced storage of the tensor.
B. Quantization and QTT format

Although some physical problems can naturally be formulated in terms of tensors and therefore solved by adapted techniques, a key point for the application of a tensor compression technique on integral equations is quantization [6], which enables a transition from a vector to a matrix to a tensor representation. For a vector \( x \in \mathbb{R}^I \) and \( I, d \in \mathbb{N} \) such that

\[
I = I_1 I_2 \ldots I_d \quad \text{where} \quad I_k \in \mathbb{N} \quad \text{and for} \quad k = 1, \ldots, d
\]
we can transform \( x \) to a \( d \)-dimensional tensor \( X \) as

\[
x_i = X(i_1, \ldots, i_d) \quad \text{with} \quad i_k = 0, 1, \ldots, I_k - 1 \quad \text{for} \quad k = 1, \ldots, d
\]
using the following flattening of the multi-index \((i_1, \ldots, i_d)\):

\[
i = i_1 + i_2 I_1 + \ldots + i_d I_1 I_2 \ldots I_{d-1}.
\]

An equivalent procedure can be applied to obtain a QTT representation for \( A \in \mathbb{R}^{I \times I} \) with quantization in both dimensions.

III. Solving Boundary Element Method with Tensor-Train Format

A. Compression with TT-cross algorithm

A tensor \( X \) in full format can be approximated by \( \tilde{X} \) in the low-rank Tensor-Train (TT) format with the TT-SVD algorithm [5] such that

\[
\|\tilde{X} - X\|_F \leq \epsilon \|X\|_F
\]
where \( \|X\|_F \) is the Frobenius norm and \( \epsilon \) is the desired accuracy.

This procedure being too expensive, it is convenient to use a TT-cross algorithm with TT-rounding [5] that is a generalization of the well-known cross algorithm used for matrices. It build

\[
\tilde{X} = f(\tilde{A}, \tilde{B}, \ldots)
\]
where \( \tilde{A}, \tilde{B} \) are cross approximations of cores \( A, B \) from the TT-SVD algorithm.

The numerical implementation is done with the use of the Python TT-Toolbox from Oseledets. The considered numerical example here is a metallic cylinder of radius 1m illuminated by an electromagnetic wave at frequency 0.6GHz. It is described with a regular mesh with \( N = 2^d \) elements. Table I gives the memory storage required by the matrix \( A \).

<table>
<thead>
<tr>
<th>( d )</th>
<th>( N )</th>
<th>Full Matrix</th>
<th>TT-cross</th>
<th>TT-rounding</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>64</td>
<td>0.0625 Mb</td>
<td>0.067 Mb</td>
<td>0.007 Mb</td>
</tr>
<tr>
<td>10</td>
<td>1024</td>
<td>16 Mb</td>
<td>0.33 Mb</td>
<td>0.048 Mb</td>
</tr>
<tr>
<td>14</td>
<td>16384</td>
<td>4 Gb</td>
<td>0.54 Mb</td>
<td>0.063 Mb</td>
</tr>
<tr>
<td>16</td>
<td>65536</td>
<td>64 Gb</td>
<td>0.7 Mb</td>
<td>0.073 Mb</td>
</tr>
<tr>
<td>18</td>
<td>262144</td>
<td>1 Tb</td>
<td>0.75 Mb</td>
<td>0.179 Mb</td>
</tr>
<tr>
<td>20</td>
<td>1048576</td>
<td>16 Tb</td>
<td>1 Mb</td>
<td>0.084 Mb</td>
</tr>
</tbody>
</table>

TABLE I
Memory Storage for the Matrix \( A \)

These results show clearly a very good compression arising from this method even for very small value of \( N \).

B. Solution using AMEN solver

The solution of the linear system may be performed by a Krylov method applying the fast vector matrix product. In this work we had applied the AMEN method developed specifically for this tensor format [7]. The results are given Table II for \( \epsilon = 0.001 \).

<table>
<thead>
<tr>
<th>( d )</th>
<th>( N )</th>
<th>full solution ( x_{ref} )</th>
<th>Amen ( x_{TT} )</th>
<th>Error ( |x_{TT} - x_{ref}| )</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>64</td>
<td>0.66</td>
<td>1.6E-11</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>256</td>
<td>1.23</td>
<td>1.09E-3</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>1024</td>
<td>2.15</td>
<td>1.78E-3</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>4096</td>
<td>3.57</td>
<td>4.5E-3</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>16384</td>
<td>5.41</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>65536</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>262144</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>1048576</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

TABLE II
Computation times (in seconds) and error with reference solution

As expected the solver make it possible to obtain very good performance due to the reduced storage and while keeping the desired accuracy.

C. Comparison with the H-Matrix techniques

We give in Table III the compression obtained on the same example with H-matrix format using an ACA technique. As a result compression and performance of the tensor solution are much higher than those obtained by H-matrix techniques. The QTT technique is known to be efficient for regular meshes and solution. The performance on more complex problems or geometries can be less efficient [4] and this tensor technique will be used as an efficient preconditioner in those cases.

REFERENCES