Report - Fast Matrix Computations

Fast Computing - Inter Semester

Groupe
BECKER Gonzalo
PISTONE WATHELET Sofía

Date d'édition : June 25, 2025
Version : 1.0

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Contents

1. Introduction

In this report, we want to study different techniques for speeding up matrix multiplication. More specifically we are going to study two methods in particular: the divide-and-conquer approach and the wavelet-based matrix sparsification. For each, a detailed explanation of the algorithm will be given, as well as some coments about their performance.

2. Divide-and-Conquer Fast Matrix Multiplication

Firstly, we will introduce the idea of divide-and-conquer fast matrix multiplication, a technique to reduce the time complexity of matrix multiplication. If the user does this operation using the traditional method we get an $O(n^3)$ where n is the size of the matrix. The main aim of this algorithm is to reduce it up to $O(n^{2,807})$.

The idea is to break down the matrix multiplication problem into smaller sub-problems and then try to solve them individually. This can be achieved by partitioning the matrices into smaller sub-matrices, then multiplying these together in parallel, and finally combining the results. Consequently, the time complexity of matrix multiplication would be reduced and ameliorate the efficiency.

2.1. Strassen Algorithm

The Strasse algorithm is a divide-and-conquer algorithm for fast matrix multiplication. This algorithm was developed by Volker Stassen in 1969 and follows the divide-and-conquer approach. The basic idea is to partition each of the matrices being multiplied into four sub-matrices, and then perform seven multiplications of these sub-matrices to obtain the product matrix. The sub-matrices are then combined to obtain the final result. This procedure can be implemented recursively which would mean that the sub-problems can be solved in parallel.

The main idea of this algorithm is that, if A, B, and C are matrices of size n x n:

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{bmatrix} \quad B = \begin{bmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \end{bmatrix} \tag{1}$$

The result of the matrix multiplication is given by:

$$C = A \cdot B = \begin{bmatrix} c_{1,1} & c_{1,2} \\ c_{2,1} & c_{2,2} \end{bmatrix} \tag{2}$$

where:

$$\begin{aligned} c_{1,1} &= a_{1,1} \cdot b_{1,1} + a_{1,2} \cdot b_{2,1} \\ c_{1,2} &= a_{1,1} \cdot b_{1,2} + a_{1,2} \cdot b_{2,2} \\ c_{2,1} &= a_{2,1} \cdot b_{1,1} + a_{2,2} \cdot b_{2,1} \\ c_{2,2} &= a_{2,1} \cdot b_{1,2} + a_{2,2} \cdot b_{2,2} \end{aligned} \tag{3}$$

In the Strassen algorithm, the calculation of C is done using 7 multiplications of 2x2 matrices, instead of the 8 multiplications used by the traditional algorithm. These 7 equations are:

$$\begin{aligned} P_1 &= a_{1,1} \cdot (b_{1,2} - b_{2,2}) \\ P_2 &= (a_{1,1} + a_{1,2}) \cdot b_{2,2} \\ P_3 &= (a_{2,1} + a_{2,2}) \cdot b_{1,1} \\ P_4 &= a_{2,2} \cdot (b_{2,1} - b_{1,1}) \\ P_5 &= (a_{1,1} + a_{2,2}) \cdot (b_{1,1} + b_{2,2}) \\ P_6 &= (a_{1,2} - a_{2,2}) \cdot (b_{2,1} + b_{2,2}) \\ P_7 &= (a_{1,1} - a_{2,1}) \cdot (b_{1,1} + b_{1,2}) \end{aligned}$$

Finally, the submatrices of C are computed as follows:

$$c_{1,1} = P_5 + P_4 - P_2 + P_6$$
$$c_{1,2} = P_1 + P_2$$
$$c_{2,1} = P_3 + P_4$$
$$c_{2,2} = P_5 + P_1 - P_3 - P_7$$

This algorithm is shown in the following flux diagram:

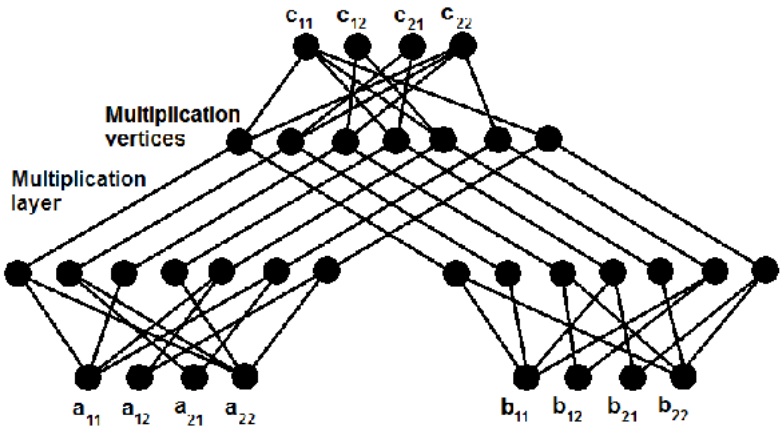

Figure 1: Strassen's algorithm for multiplying two 2 × 2 matrices A and B.

The outcome is that the number of scalar multiplication required to perform a matrix multiplication is reduced in comparison with the standard method. Consequently, the computation time is reduced and the efficiency is increased.

2.2. Non-commutative

The Strassen algorithm is non-commutative, meaning it does not depend on the commutative property of the multiplication. This is important to note due to the fact that noncommutative algorithms may be applied as it is to block matrices. In other words we could grab the $a_{i,j}, b_{i,j}$ and $c_{i.j}$ in the equations previously mentioned and replace them with block matrices and it would still work.

2.3. Arbitrary Precision Approximation (APA) Algorithms

APA Algorithms are noncommutative algorithms for 2x2 and 3x3 matrix multiplication, and it requires even $less$ equations than the Strassen method. The price is that this method requires longer word lengths.

The algorithm uses the following equation to perform the multiplication $A\cdot = C$:

Let given A and B previously mentioned, then the product C = $A\cdot$ is given by:

$$c_{1,1} = a_{1,1}b_{1,1} + a_{1,2}b_{2,1}$$
$$c_{1,2} = a_{1,1}b_{1,2} + a_{1,2}b_{2,2}$$
$$c_{2,1} = a_{2,1}b_{1,1} + a_{2,2}b_{2,1}$$
$$c_{2,2} = a_{2,1}b_{1,2} + a_{2,2}b_{2,2}$$

These 4 basic equations allow the user to obtain the multiplication of two 2x2 matrices, this makes this algorithm an efficient and accurate method for this operation.

2.4. Number Theoretic Transform (NTT) based algorithm

Similar to APA algorithm, but more flexible, there is an algorithm called a number Theoretic Transform (NTT) based algorithm. In this method, matrix multiplication is reformulated as a linear convolution,

which can be thought of as the multiplication of two polynomials using the z-transform. Then, the result is transformed back into a matrix.

This process involves several steps:

1. Padding: The first step is to pad the matrices to be multiplied with zeros to reach a size that is a power of two. This is done to ensure that the NTT algorithm can be applied efficiently.
2. Scaling: This are scaled by a factor of N.
3. Number Theoretic Transform

If A and B are two NxN matrices to be multiplied and C is the result, this algorithm calculates the elements of C as:

$$C_{i,j} = \sum_{k=1}^{N} a_{i,k} b_{k,j}$$

To implement the NTT algorithm, we first pad the matrices with zeros to reach a size that is a power of two. Now, N' is the padded size. The padded matrices are then scaled by a factor of N':

$$A_{i,j} = \frac{A_{i,j}}{N'} \qquad B_{i,j} = \frac{B_{i,j}}{N'}$$

Later, the NTT of each element is calculated using the formula:

$$A'i,j = \sum_{i=0}^{N'-1} \sum_{j=0}^{N'-1} a_{i+jN} x^{i+jN} \qquad B'i,j = \sum_{i=0}^{N'-1} \sum_{j=0}^{N'-1} b_{N-1-i+jN} x^{N(N-1-i+jN)}$$

Finally, the elements of the product matrix are computed as follows:

$$C_{i,j} = \sum_{k=1}^{N'} A'i,kB'k,j = \sum_{i=1}^{N'^3+N'^2-N-1} c_i x^i$$

The NTT algorithm is efficient because it takes advantage of the properties of modular arithmetic and number theory to perform polynomial multiplication more quickly than other methods, such as traditional polynomial multiplication. This is useful in optical computing architectures for multiplying large numbers.

3. Wavelet-Based Matrix Sparsification

3.1. Introduction

Wavelet-Based Matrix Sparsification, also known as Wavelet Galerkin method, is a method for computing integral equations. By applying specific transformations (such as Galerkin's method), the integral equation can be changed into a linear system, where each coefficient is associated with a specific basis function. Therefore, by making an optimal choice of the basis functions, a sparse representation can be achieved. A sparse representation strongly improves the efficiency of matrix-vector products required for the iterative method that will solve the system. The key is that wavelets have nice compression properties, which allow decomposing a function with relatively sparse coefficients.

3.2. Wavelet Transform

The wavelet transform can be represented mathematically using the following equation:

$$Wf(u,s) = \frac{1}{\sqrt{s}} \int_{-\infty}^{\infty} f(t)\ \psi^* \left(\frac{t-u}{s}\right) dt$$

where $Wf(u,s)$ is the wavelet coefficient, $f(t)$ is the original signal, $\psi(t)$ is the wavelet function, s is the scale factor, and u is the translation factor. The wavelet function $\psi(t)$ is a special type of function called a wavelet mother, which is used to generate a set of wavelets with different scales and translations. Particularly, the following family of wavelets, that composes an orthonormal basis of $L^2(\mathbb{R})$, is used in this development:

$$\left\{\psi_{j,n}(t) = \frac{1}{\sqrt{2^j}}\psi\left(\frac{t-2^j n}{2^j}\right)\right\}_{(j,n)\in\mathbb{Z}^2}$$

Finally, a representation of $f(t)$ can be obtained, where $\phi(x)$ is the scaling function:

$$f(x) = \sum_k c_k\ \phi(2^j x - k) + \sum_k d_k\ \psi(2^j x - k)$$

Scaling functions are used to capture the coarse structure of the signal, while wavelets capture the fine structure or details of the signal. This is because scaling functions have a smooth frequency response, while wavelets posses a narrow one. In other words, the scaling functions provide a low-pass filtered version of the signal, while the wavelets provide high-pass filtered versions of the signal that are sensitive to local changes in the signal.

To implement this is the discrete world, we must compute the coefficients of the representation. For that purpose, the Mallat algorithm is a fast wavelet transform discrete algorithm that provides an efficient way to compute the discrete wavelet transform, which is a representation of a signal in the wavelet domain. The key idea behind the Mallat algorithm is to use a filter bank to decompose a signal into different frequency bands, and then to perform a downsampling operation to reduce the size of the data at each stage. This process is repeated until the desired level of detail is reached.

3.3. Galerkin method

The Galerkin method is a numerical technique for solving integral equations with a known kernel. It involves approximating the solution of the integral equation as a finite linear combination of wavelets. In general, $K(x,y)$ and $g(x)$ are known in the following integral equation, and $f(y)$ is what we are looking for.

$$g(x) = \int_{-\infty}^{\infty} K(x,y)\ f(y)\ dy$$

By using the expression obtained in the previous section to express $f(y)$ and $g(x)$, and by using the 2D wavelet for $K(x,y)$, cross terms will be canceled due to the orthogonality of the basis functions, obtaining therefore the following relationship:

$$\sum_j \sum_n K_{i,j}(m,n) f_j(n) = g_i(m)$$

The coefficients of the matrix can then be computed by using the fast wavelet algorithm of Mallat.

3.4. Sparsity

A wavelet has vanishing moments of order m if, for $n = 1, 2, ..., m$:

$$\int_{-\infty}^{\infty} x^n \psi(x) dx = 0$$

If this equation stands, it can be proven that, under specific conditions, the 2D development of $K(x,y)$ using wavelets and scaling functions will give a sparse matrix. In effect, if $|m-n|$ is the row-column difference for a specific coefficient, its value will be attenuated by approximately $|m-n|^{k+1}$ for some $k > 1$. By imposing a threshold on the minimum value of each coefficient, a highly sparse matrix results.

This method is effective in achieving a sparse matrix to represent the original equation, as it can be seen in the following image, that shows a matrix that resulted from this method, obtained from [4]:

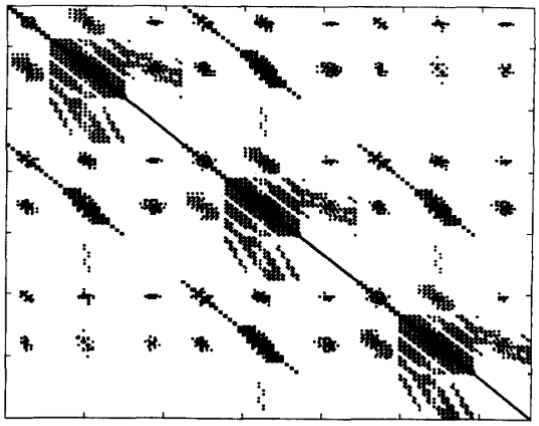

Fig. 6. The structure of moment matrix of geometry of Fig. 5 after applying a threshold of 1%.

3.5. Performance

Although effective, this method can present some disadvantages. For instance, there are a number of factors that could affect its performance, such as approximation errors caused by the wavelet basis used in the method and potential numerical instability. Nonlinear problems can also be challenging for the wavelet Galerkin method to handle, and it may be computationally intensive for high-dimensional problems. While the method can be efficient in some cases, its performance will depend on the specific problem being solved, and it may not always be the best choice.

Regarding its complexity, it can be shown that the number of operations required to do the matrix computation varies between $O(N)$ (under many restrictive assumptions and neglecting certain calculations) and $O(N^3)$. More in detail, the reduction in order can be achieved by neglecting scale interactions, and normally the complexity will be around $O(Nlog(N)^2)$. The loss of accuracy implies however a difficult trade-off concerning this method.

All in all, it should be noted that the choice of this algorithm requires an elevated comprehension of its complexity, accuracy and robustness in the context of the specific problem being solved.

4. Bibliography

References

[1] "Fast Matrix Computations" Digital Signal Processing Handbook, Yagle, A.E.

[2] A Parallel Structured Divide-and-Conquer Algorithm for Symmetric Tridiagonal Eigenvalue Problems, Xia Liao, Shengguo Li , Yutong Lu, and Jose E. Roman.

[3] A Wavelet Tour of Signal Processing, Stéphane Mallat

[4] Analysis of Integrated Millimeter-Wave and Submillimeter- Wave Waveguides, Kazem Sabetfakhri and Linda P. B. Katehi

[5] Wavelet Galerkin method for solving singular integral equations, K. MALEKNEJAD, M. NOSRATI and E. NAJAFI Using Orthonormal Wavelet Expansions

[6] Computational aspects of harmonic wavelet Galerkin methods and an application to a precipitation front propagation model, Saulo R.M. Barros , Pedro S. Peixoto