ELEN4020-LAB1

Emma Clark
Jason Smit
Isabel Tollman
School of Electrical & Information Engineering.
University of the Witwatersrand
Private Bag 3, 2050, Johannesburg, South Africa

Abstract: This report presents the procedure for tensor multiplication and addition of rank 2 and 3.

Key words: 2D, 3D, Tensor, pseudocode

1. INTRODUCTION

This report details the procedure for addition and multiplication of Tensors of rank 2D and 3D. For each section the procedure is outlined and pseudo code is presented. Each of the procedures is implemented using C++.

2. 2D TENSOR ADD

The addition of two 2-dimensional matrices is achieved using element-by-element addition, as shown in Equation 1. Algorithm 1 displays the pseudo-code used to calculate the addition of two N×N matrices.

$$C_{ij} = A_{ij} + B_{ij} \tag{1}$$

2.1 Code

```
Algorithm 1: rank2TensorAdd finds the addition of two $N \times N$ matrices
  Input: Three $N \times N$ constant integer arrays: $a[N][N], b[N][N]$ and $c[N][N]$
  Output: void
1 $N \leftarrow$ size($a$)
2 for $i \leftarrow 0$ to $N-1$ do
3     for $j \leftarrow 0$ to $N-1$ do
4         $c[i][j] = a[i][j] + b[i][j]$
```

3. 2D TENSOR MULTIPLY

The multiplication of two 2-dimensional matrices is achieved by performing the dot product on the respective rows and columns, as illustrated in Equation 2. Algorithm 2 shows the pseudo-code used to multiply two N×N matrices.

$$C_{ij} = \sum_{k} A_{ij} \times B_{ij} \tag{2}$$

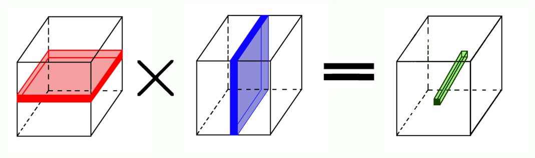

Figure 1 : Visualisation of a tensors multiplication of rank 3

3.1 Code

Algorithm 2: rank2TensorMulti finds the multiplication of two $N \times N$ matrices

```
Input: Three $N \times N$ constant integer arrays: $a[N][N], b[N][N]$ and $c[N][N]$
Output: void
1 $N \leftarrow$ size$(a)$
2 for $k \leftarrow 0$ to $N-1$ do
3     for $i \leftarrow 0$ to $N-1$ do
4         for $j \leftarrow 0$ to $N-1$ do
5             $c[i][j] = c[i][j] + a[i][j] \times b[i][j]$
```

4. 3D TENSOR ADD

The addition of two 3-dimensional arrays is achieved using element-by-element addition,similar to a 2D array. The difference is only the addition of the inclusion of the additional dimension. Algorithm 3 shows the pseudo-code used to sum two N×N×N matrices.

4.1 Code

Algorithm 3: rank3TensorAdd finds the addition of two $N \times N \times N$ matrices

```
Input: Three $N \times N \times N$ constant integer arrays: $a[N][N][N], b[N][N][N]$ and $c[N][N][N]$
Output: void
1 $N \leftarrow$ size$(a)$
2 for $k \leftarrow 0$ to $N-1$ do
3     for $i \leftarrow 0$ to $N-1$ do
4         for $j \leftarrow 0$ to $N-1$ do
5             $c[i][j][k] = a[i][j][k] + b[i][j][k]$
```

5. 3D TENSOR MULTIPLY

The multiplication of two 3-dimensional arrays makes use of 2-dimensional matrix multiplication as a basis. The i^{th}row-plane of array A and the j^{th} column-plane of array B are multiplied using traditional 2-dimensional matrix multiplication shown in Algorithm 2. The result is the k^{th} layer-plane of array C. Algorithm 4 shows the pseudo-code used to multiply two N×N×N matrices. This can be visualised as is seen in figure 1

5.1 Code

Algorithm 4: rank3TensorMulti finds the multiplication of two $N \times N \times N$ matrices

```
Input: Three $N \times N \times N$ constant integer arrays: $a[N][N][N], b[N][N][N]$ and $c[N][N][N]$
Output: void
1 $N \leftarrow \text{size}(a)$
2 $temp\_c = zeros(N, N)$
3 for $k \leftarrow 0$ to $N - 1$ do
4     $temp\_a = a[k][:][:]$
5     $temp\_b = b[:][k][:]$
6     $rank2TensorMulti(temp\_a[N][N], temp\_b[N][N], temp\_c[N][N])$
7     $c[:][:][k] = temp\_c$
```

6. IMPLEMENTATION

The preceding sections' pseudocode is implemented using c++. The code generated performs as detailed by the pseudocode and is accessible via the public github link `https://github.com/IsabelTollman/ELEN4020A_Group7/tree/Code` under the '*Code*' branch.

7. CONCLUSIONS

[?]

REFERENCES