CAAM 419/519, Homework #3

hc54

November 21, 2022

1 Diagonal Matrix

1.1 Output of print

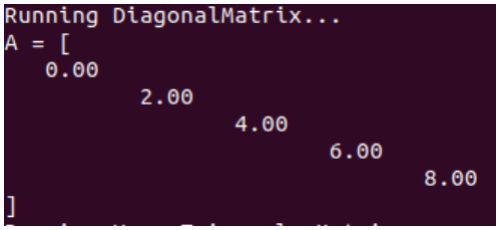

Figure 1: 5-by-5 Diagonal Matrix

1.2 Verification of the Correctness

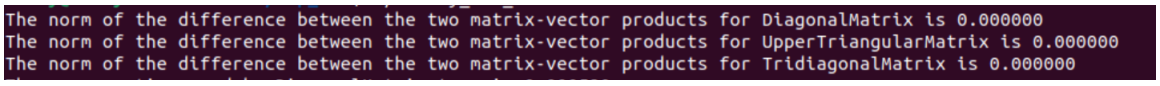

Figure 2: Verification

The output 0 is the norm of the difference between output vector from multiply_Matrix_Vector and output vector from multiply_DiagonalMatrix_Vector.

1.3 Discussion of Implementing Function

```
void multiply_DiagonalMatrix_Vector(Vector* out, DiagonalMatrix* A, Vector* x){
    for (int i = 0; i < A->n; ++i){
         out->ptr[i] = 0.0;
         double A_i = A->ptr[i];
         double x_i = x->ptr[i];
         out->ptr[i] += A_i * x_i;
         }
    }
```

Suppose we have a given vector x with length n, and a n-by-n matrix A. The output vector is *out* with length n. A_{ii} is the diagonal entry of row i.

$$out_i = A_{ii} \times x_i \textbf{ for i = 1...n}$$

```
The average time used by DiagonalMatrix type is 7.3e-07 s
The average time used by Matrix type is 4.564e-05 s
```

Figure 3: Runtime Comparison

1.4 Comparison of the Runtime

We can observe that the running time of function with DiagonalMatrix type is a lot faster than the function with Matrix type. The one with Matrix type compute approximately 100×100 times if n = 100, but the one with DiagonalMatrix type compute approximately 100 times, which is faster by roughly factor of 100.

2 Upper Triangular Matrix

2.1 Output of print

```
Running UpperTriangularMatrix...
A = [
   0.00   1.00   2.00   3.00   4.00
          2.00   3.00   4.00   5.00
                 4.00   5.00   6.00
                        6.00   7.00
                               8.00
]
```

Figure 4: 5-by-5 Upper Triangular Matrix

2.2 Verification of the Correctness

```
The norm of the difference between the two matrix-vector products for DiagonalMatrix is 0.000000
The norm of the difference between the two matrix-vector products for UpperTriangularMatrix is 0.000000
The norm of the difference between the two matrix-vector products for TridiagonalMatrix is 0.000000
```

Figure 5: Verification

The output 0 is the norm of the difference between output vector from multiply_Matrix_Vector and output vector from multiply_UpperTriangularMatrix_Vector.

2.3 Discussion of Implementing Function

```
void multiply_UpperTriangularMatrix_Vector(Vector* out, UpperTriangularMatrix* A,
Vector* x){
     for (int i = 0; i < A->n; ++i){
           out->ptr[i] = 0.0;
           for (int j = i; j < A->n; ++j){
               double A_ij = A->ptr[i][j];
               double x_j = x->ptr[j];
               out->ptr[i] += A_ij * x_j;
            }
        }
    }
```

Suppose we have a given vector x with length n, and a n-by-n matrix A. The output vector is *out* with length n.

$$out_i = \sum_{j=i}^{n} A_{ij} \times x_i \textbf{ for i = 1,...,n}$$

2.4 Comparison of the Runtime

```
The average time used by UpperTriangularMatrix type is 2.493e-05 s
The average time used by Matrix type is 4.846e-05 s
```

Figure 6: Runtime Comparison

We can observe that the running time of function with UpperTriangularMatrix type is faster than the function with Matrix type. The one with Matrix type compute approximately 100×100 times if n = 100, but the one with DiagonalMatrix type compute approximately (100×101)/2 times, which is faster by roughly factor of 2.

3 Tridiagonal Matrix

3.1 Output of print

```
Running TridiagonalMatrix...
A = [
   0.00   1.00
   1.00   2.00   3.00
          3.00   4.00   5.00
                 5.00   6.00   7.00
                        7.00   8.00
]
```

Figure 7: 5-by-5 Tridiagonal Matrix

3.2 Verification of the Correctness

```
The norm of the difference between the two matrix-vector products for DiagonalMatrix is 0.000000
The norm of the difference between the two matrix-vector products for UpperTriangularMatrix is 0.000000
The norm of the difference between the two matrix-vector products for TridiagonalMatrix is 0.000000
```

Figure 8: Verification

The output 0 is the norm of the difference between output vector from multiply_Matrix_Vector and output vector from multiply_TridiagonalMatrix_Vector.

3.3 Discussion of Implementing Function

```
void multiply_TridiagonalMatrix_Vector(Vector* out, TridiagonalMatrix* A, Vector* x){
    for (int i = 0; i < A->n; ++i){
        out->ptr[i] = 0.0;
        if (i==0){
           out -> ptr[i] = A->ptr_mid[i]*x->ptr[i] + A->ptr_upper[0]*x->ptr[1];
           }
        else if (i>0 && i < (A->n)-1){
           out ->ptr[i] = A->ptr_mid[i]*x->ptr[i] + A->ptr_lower[i-1]*x->ptr[i-1]
           + A->ptr_upper[i]*x->ptr[i+1];
           }
        else if (i == (A->n)-1){
           out -> ptr[i] = A->ptr_mid[i]*x->ptr[i] + A->ptr_lower[i-1]*x->ptr[i-1];
           }
```

```
}
}
```

Suppose we have a given vector x with length n, and a n-by-n matrix A. The output vector is *out* with length n.

$$out_i = A_{i1} \times x_1 + A_{i2} \times x_2 \textbf{ for i = 1}$$

$$out_i = A_{ii} \times x_i + A_{i,i-1} \times x_{i-1} + A_{i,i+1} \times x_{i+1} \textbf{ for i = 2,...,n-1}$$

$$out_i = A_{i,n} \times x_n + A_{i,n-1} \times x_{n-1} \textbf{ for i = n}$$

3.4 Comparison of the Runtime

```
The average time used by Tridiagonal type is 8.9e-07 s
The average time used by Matrix type is 4.094e-05 s
```

Figure 9: Runtime Comparison

We can observe that the running time of function with TridiagonalMatrix type is a lot faster than the function with Matrix type. The one with Matrix type compute approximately 100×100 times if n = 100, but the one with TridiagonalMatrix type compute approximately 100×3 times, which is faster by roughly factor of 100.