CAAM 519, Homework #3

ask15

November 16, 2022

1 Matrix

1.1 Output of matrix print method

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

2 Diagonal Matrix

2.1 Output of matrix print method

```
Running Diagonal Matrix...
A = [
    0.00
            2.00
                    4.00
                            6.00
                                    8.00
]
```

2.2 Correctness of the matrix-vector multiplication implementation

```
The norm of the difference between the two matrix-vector products
for Diagonal Matrix is 0.000000
```

2.3 Implementation of the matrix-vector multiplication

For the diagonal matrix, since we only have the diagonal entries stored, we can directly multiply element-wise the diagonal elements by the entries of the vector.

2.4 Comparison of the runtime

```
The average run time for matrix_vector multiplication
for Diagonal matrix is 0.000000530
The average run time for matrix_vector multiplication
for Diagonal matrix using Matrix type is 0.000071380
```

For n=100, the average run time for matrix vector multiplication for diagonal matrix (custom type) is around 100 times faster than that using the regular matrix type since we are only storing the diagonal entries (100 elements) instead of 100x100 elements of the matrix and using them in the computations.

3 Upper Triangular Matrix

3.1 Output of matrix print method

```
Running Upper Triangular Matrix...
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
]
```

3.2 Correctness of the matrix-vector multiplication implementation

```
The norm of the difference between the two matrix-vector products
for Upper Triangular Matrix is 0.000000
```

3.3 Implementation of matrix-vector multiplication

Since we are storing all elements of the upper triangular matrix in a row major fashion in a one dimensional array, we can utilize a counter to access the elements. For each row, we do a dot product between the elements of the row and the corresponding elements of the vector.

3.4 Comparison of the runtime

```
The average run time for matrix_vector multiplication
for Upper Triangular matrix is 0.000034500
The average run time for matrix_vector multiplication
for Upper Triangular matrix using Matrix type is 0.000075270
```

For n=100, the average run time for matrix vector multiplication for upper triangular matrix (custom type) is around 2 times faster than that using the

regular matrix type since we are effectively storing only the upper half of the matrix and using it in our computations.

4 Tridiagonal Matrix

4.1 Output of matrix print method

```
Running Tridiagonal Matrix...
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
]
```

4.2 Correctness of the matrix-vector multiplication implementation

```
The norm of the difference between the two matrix-vector products
for Tridiagonal Matrix is 0.000000
```

4.3 Implementation of matrix-vector multiplication

For the tridiagonal matrix, we had the first and final elements of the product vector done first as they need to only access 2 elements of the tridiagonal matrix rather than 3. For the rest, we have a pattern to do the dot product between the 3 elements and their corresponding elements of the vector.

4.4 Comparison of the runtime

```
The average run time for matrix_vector multiplication
for Tridiagonal matrix is 0.000000960
The average run time for matrix_vector multiplication
for Tridiagonal matrix using Matrix type is 0.000095510
```

For n=100, the average run time for matrix vector multiplication for upper triangular matrix (custom type) is around 100 times faster than that using the regular matrix type since we are effectively storing 3 diagonals of around 100 elements instead of 100x100 elements and using it in our computations.