Program construction in C++ for Scientific Computing Teacher: Michael Hanke

Ilian Häggmark
mail ilianh@kth.se

Andreas Karlsson
mail andreas.a.karlsson@ki.se

October 24, 2019

1 Project 2

1.1 Task 1 - implement the evaluation of the exponential for real numbers.

The exponential for real numbers are based on a simple Taylor expansion and the implementation uses Horner's scheme to improve the performance. To estimate the error, a for-loop calculate each of the terms in the Taylor expansion. If term n is smaller than the error the sum of consecutive terms is smaller than the term n and thus the error. Horner's scheme then uses n-terms for the approximation.

tol \ x	-1	1	3	5	10	50
0.01	0.00121277	0.00161516	0.00143384	0.00294883	0.00175855	3.14573e+06
0.001	2.22983e-05	2.78602e-05	6.83293e-05	0.00020803	0.000162384	3.14573e+06
1e-08	1.49839e-10	1.72876e-10	1.67e-09	8.31676e-10	9.24047e-10	3.14573e+06
1e-10	7.19536e-13	8.15348e-13	4.16023e-12	4.17799e-12	1.45519e-11	3.14573e+06

Table 1: Absolute difference between the cmath implementation and our implementation of an exponential function using Taylor expansion and Horner's scheme for a number of x values and tolerances. 1.

The code is tested with a set of tolerances and exponents x. As can be seen in Table 1 the absolute difference with the `cmath exp` function is within the set tolerance for low x values. However for x larger than 10 the error exceeds the set tolerance. This is because the type `double` cannot store data with high enough precision. A `double` can store about 15 digits. If the desired error should be 10^{-10} then the largest value that can be stored with this precision is $\sim 10^5$. Using libraries that can handle numbers with higher precision, e.g. boost or GMP, might increase the accuracy of our current implementation. To further improve the accuracy we could also investigate the use of one of the other four classes of methods for calculating the exponential of a matrix described by Moler and Van Loan [2003].

1.2 Task 2 - construct a matrix class.

A basic matrix class has been constructed that contains the basic constructors and operator overloading that are required to calculate the matrix exponential with Taylor expansion and Horner's scheme, i.e., the same way as in Section 1.1.

The matrix object contains an `int` with the size of the matrix and a C++ vector object that contains the matrix values. The approach to determine the required number of terms in the Taylor expansion to get an error below a set tolerance is equivalent to that in Section 1.1.

In order to test the `matrixExp` function we compared it with Matlab's `r8mat_expm1` function. We started by creating a `Matrix` class objects with low values called $M_{\text{input}}^{\text{low}}$.

$$M_{\text{input}}^{\text{low}} = \begin{bmatrix} 1 & 3 & 1 & 4 \\ 2 & 3 & 1 & 0 \\ 2 & 1 & 3 & 2 \\ 4 & 4 & 2 & 1 \end{bmatrix}$$

We then used our `matrixExp` function to calculate the exponential of the matrix $E_{\text{our}}^{\text{low}}$.

$$E_{\text{our}}^{\text{low}} = \texttt{matrixExp}(M_{\text{input}}^{\text{low}}, 0.001) = \begin{bmatrix} 1073.04 & 1411.07 & 795.17 & 820.404 \\ 608.444 & 802.657 & 451.507 & 463.959 \\ 1001.2 & 1310.54 & 748.678 & 768.065 \\ 1216.89 & 1599.88 & 903.013 & 930.636 \end{bmatrix}$$

We also calculated the exponential of the matrix using Matlab's `r8mat_expm1` function, which produces the matrix displayed as $E_{\text{matlab}}^{\text{low}}$.

$$E_{\text{matlab}}^{\text{low}} = \texttt{r8mat\_expm1}(4, \texttt{mArray}) = \begin{bmatrix} 1073.04 & 1411.07 & 795.17 & 820.404 \\ 608.444 & 802.657 & 451.507 & 463.959 \\ 1001.2 & 1310.54 & 748.678 & 768.065 \\ 1216.89 & 1599.88 & 903.013 & 930.636 \end{bmatrix}$$

The two methods produced very similar results as can be concluded from the calculated difference which is displayed as $E_{\text{difference}}^{\text{low}}$.

$$E_{\text{difference}}^{\text{low}} = E_{\text{our}}^{\text{low}} - E_{\text{matlab}}^{\text{low}} = \begin{bmatrix} -4.85429e-05 & -6.38154e-05 & -3.60653e-05 & -3.71208e-05 \\ -2.75338e-05 & -3.61964e-05 & -2.04564e-05 & -2.10551e-05 \\ -4.53725e-05 & -5.96475e-05 & -3.37098e-05 & -3.46963e-05 \\ -5.50676e-05 & -7.23928e-05 & -4.09128e-05 & -4.21102e-05 \end{bmatrix}$$

To investigate how well our function `matrixExp` could handle larger values we then created a `Matrix` class objects called $M_{\text{input}}^{\text{high}}$ with higher values.

$$M_{\text{input}}^{\text{high}} = \begin{bmatrix} 1 & 3 & 10 & 45 \\ 12 & 3 & 5 & 0 \\ 12 & 1 & 3 & 7 \\ 19 & 4 & 9 & 6 \end{bmatrix}$$

We then used our `matrixExp` function to calculate the exponential of the matrix $E_{\text{our}}^{\text{high}}$.

$$E_{\text{our}}^{\text{high}} = \texttt{matrixExp}(M_{\text{input}}^{\text{high}}, 0.001) = \begin{bmatrix} 8.47678e+16 & 2.18816e+16 & 5.62372e+16 & 1.24171e+17 \\ 3.30298e+16 & 8.52616e+15 & 2.19129e+16 & 4.83831e+16 \\ 4.02549e+16 & 1.03912e+16 & 2.67062e+16 & 5.89667e+16 \\ 6.21118e+16 & 1.60332e+16 & 4.12066e+16 & 9.09833e+16 \end{bmatrix}$$

We also calculated the exponential of the matrix using Matlab's `r8mat_expm1` function, which produces the matrix displayed as $E_{\text{matlab}}^{\text{high}}$.

$$E_{\text{matlab}}^{\text{high}} = \texttt{r8mat\_expm1}(4, \texttt{mArray}) = \begin{bmatrix} 8.47678e+16 & 2.18816e+16 & 5.62372e+16 & 1.24171e+17 \\ 3.30298e+16 & 8.52616e+15 & 2.19129e+16 & 4.83831e+16 \\ 4.02549e+16 & 1.03912e+16 & 2.67062e+16 & 5.89667e+16 \\ 6.21118e+16 & 1.60332e+16 & 4.12066e+16 & 9.09833e+16 \end{bmatrix}$$

We then compared the two functions by calculating the difference which is displayed as $E_{\text{difference}}^{\text{high}}$.

$$E_{\text{difference}}^{\text{high}} = E_{\text{our}}^{\text{high}} - E_{\text{matlab}}^{\text{high}} = \begin{bmatrix} 672 & 144 & 448 & 784 \\ 208 & 45 & 140 & 248 \\ 320 & 70 & 216 & 400 \\ 424 & 92 & 288 & 512 \end{bmatrix}$$

In analog to Section 1.1 our `matrixExp` function performed well for matrices containing elements with low values and as we increased the values of the matrix elements we lose precision. This resulted in the observed difference between our implementation and MatLab's `r8mat_expm1` function. Potential remedies for this were discussed in Section 1.1.

References

Cleve Moler and Charles Van Loan. Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. 45(1):3–49, March 2003. ISSN 0036-1445 (print), 1095-7200 (electronic). doi: http://dx.doi.org/10.1137/S00361445024180. URL `http://epubs.siam.org/sam-bin/dbq/article/41801`.