# Illustrative examples

In this section, we will study the performance and effectiveness of the proposed algorithm (Algorithm 2.1) for three different numerical examples. All experiments were performed in Matlab R2019a using Inter(R) Core(TM) i5-8300H CPU processor. Meanwhile, we use the MATLAB stopwatch timer functions, tic and toc, to measure the performance of the algorithms.

**Example 3.1**. In this example,we compute the determinant of a 100-by-100 heptadiagonal Toeplitz matrix. It shows the intermediate process and the final result of calculating the determinant using the algorithm we proposed above。

T

By using Algorithm (2.1) 删除原来的这个->（with k=4）<- obtain

A=[] B=[] C=[]

If we set k=10,并且r=(100-6) mod k= 4,q=(100-6-r)/k=9,E=A,C=D^k,

next,

For i = 1,2,···9

compute E= CE

End

For j = 1,2,··· ,r,

compute E = DE

End

Then we can obtain：

E=[]

and

P=[]

Finally,we have

det(T100)=(-1)^(100-1)\*d^(100-6)\*det(P)=3.402823669209385e + 38

**Example 3.2.** Next,we compute the determinant of an n-by-n heptadiagonal Toeplitz matrix.

T=[]

We have tested and compared the computing time when using different approaches to compute the determinent of matrix with low order. Let n be 10, 30, 100, 200, 500, 1000, 2000, 4000 one by one.The results with Gauss elimination method,Matlab built-infunction,and our proposed algorithm (Algorithm 2.1) are given in Table 1.Figure 1 shows the runing time of CPU times(average values after 50 tests) in three algorithms.

Table 1 Numeical results of the determinants for Example 3.2

Figure 1 CPU times (after 50 times), in log scale for Example 3.2

From Table 1,we note that our algorithm generated the same values as the others.Meanwhile,Figure 1 shows that the CPU times (mean values after 50 tests)of our algorithm are less than those of other algorithms. Although the CPU times returned by these functions are not always objective , it can reflect the relative speed of our different algorithms。

**Example 3.3.** In this example, we begin to compute some determinents with high order. Let n be 6000, 9000, 12000, 15000, 18000, 21000, 24000 one by one.

T=[]

Table 1 Numeical results of the determinants for Example 3.3

Figure 1 CPU times (after 50 times), in log scale for Example 3.3

For different n, Table 2 shows us the results of three different methods. The result of Algorithm 2.1 is same with the result of MATLAB. Figure2 shows us the average time of running time of CPU,which suggests that Algorithm.1 is far more effecient than MATLAB and Gauss elimination method. Moreover, for different orders given above, there is no obvious increase trend in running time in algorithm 3.2. Besides, there are many division operations in Gauss elimination method, making it easy to break down when using it. However, there is no need to worry about it when using our method.