1.

(h)

\							
\ method	LU	banded	sparse LU	R	banded R	sparse R.	chol
n \							
200	3.97ms	0.33ms	0.51ms	0.54ms	0.28ms	15.76ms	1.80ms
400	8.14ms	0.33ms	1.31ms	1.20ms	0.36ms	18.78ms	1.50ms
600	15.19ms	0.35ms	0.55ms	0.90ms	0.30ms	24.97ms	3.14ms
800	27.23ms	0.32ms	0.62ms	1.67ms	0.34ms	32.73ms	7.31ms
1000	38.94ms	0.35ms	0.71ms	3.44ms	0.42ms	40.54ms	11.98ms
1200	55.67ms	0.40ms	1.81ms	4.67ms	0.72ms	49.88ms	16.38ms
1400	69.50ms	0.42ms	1.72ms	6.05ms	0.48ms	56.29ms	22.86ms
1600	102.67ms	0.45ms	1.15ms	6.58ms	0.50ms	63.17ms	34.25ms

(i)

1) the two banded methods(banded LU in b and banded prefactored in e) seems to be more efficient than others.

2) banded LU and banded R solves the linear system faster than normal LU and normal R, especially when array size is large, because banded matrix only takes non-zero terms, so they have smaller space complexity, it takes less time to solve the problem.

3) spSolve is similar to banded one, it only takes in non-zero entries of the matrix, when n is large, the efficiency becomes much larger by factor n**2, solving the system therefore takes significant less time.

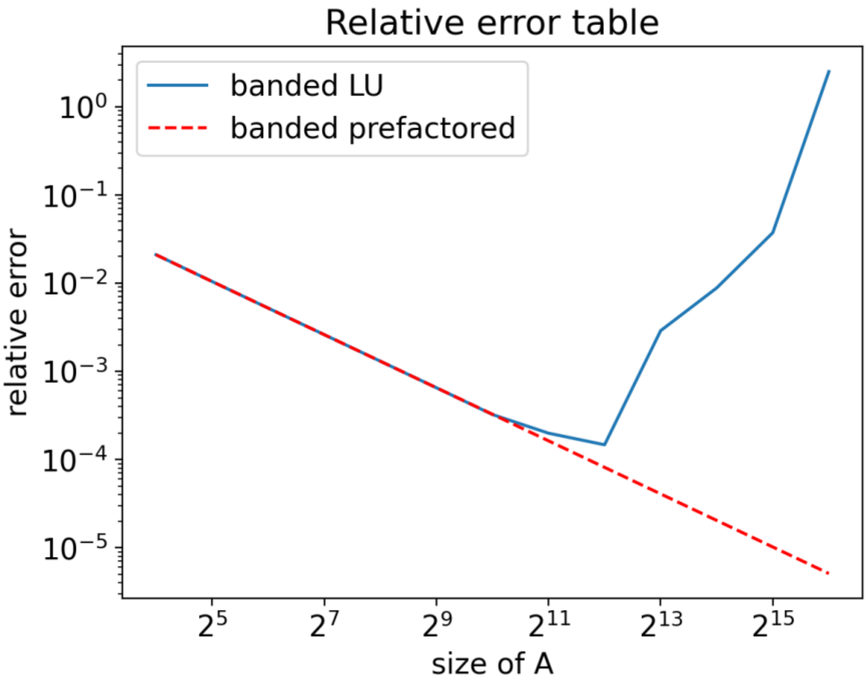
Sparse prefactored runs extremely slower than others, that’s maybe because that csr_matrix stores the matrix by indexes of row and column, which can be very slow in spsolve_triangular due to memory inefficiency, because solve triangular requires row-based and column-based operations.

4) the prefactored linear system takes less time than LU, especially when n is large,

because it turn the matrix to upper or lower triangular, so that they can use forward or back substitution, which has less complexity than solving linear system directly.

5) Cholesky runs faster than LU decomposition because it recognizes A as Hermitian positive-definite matrix. The algorithm decomposes A as LL* or U*U, then solve Ly = b and L.Hx = y, which shows efficiency when n is large.

2.



B) As shown in the figure, relative error of banded_prefactored tends to get smaller linearly when array size getting larger, while for banded LU, the error stays for the same as banded prefactored from 16 to around 1000, but the error becomes larger and unstable when n is getting bigger.

The difference in relative error of these two algorithms is caused by the different form of the matrixes. Banded prefactored takes in two triangular matrixes, which has two linear systems that is easier to solve by substitutions. But the banded LU is not upper or lower triangular, so they need complicated row reduction especially when array size is large, when doing addition or subtraction, cancellation error may occur if two numbers differences a lot. And, the program may overflow or underflow if the result of multiplication or division is too large or too small. So when n is getting larger, the error of banded LU gets larger.

Relative error of banded prefactored gets smaller linearly when n gets larger. Which is because true value doesn’t change, when array size gets larger, the infinity norm of array gets more accurate to the true value. The relative error = (appro – true)/true, should get smaller respectively.