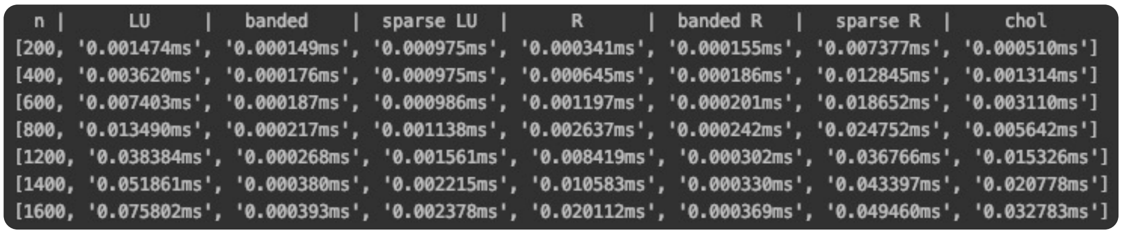
1. w)



```
 n |      LU      |   banded     |  sparse LU  |      R       |  banded R   |  sparse R  |     chol
[200, '0.001474ms', '0.000149ms', '0.000975ms', '0.000341ms', '0.000155ms', '0.007377ms', '0.000510ms']
[400, '0.003620ms', '0.000176ms', '0.000975ms', '0.000645ms', '0.000186ms', '0.012845ms', '0.001314ms']
[600, '0.007403ms', '0.000187ms', '0.000986ms', '0.001197ms', '0.000201ms', '0.018652ms', '0.003110ms']
[800, '0.013490ms', '0.000217ms', '0.001138ms', '0.002637ms', '0.000242ms', '0.024752ms', '0.005642ms']
[1200, '0.038384ms', '0.000268ms', '0.001561ms', '0.008419ms', '0.000302ms', '0.036766ms', '0.015326ms']
[1400, '0.051861ms', '0.000380ms', '0.002215ms', '0.010583ms', '0.000330ms', '0.043397ms', '0.020778ms']
[1600, '0.075802ms', '0.000393ms', '0.002378ms', '0.020112ms', '0.000369ms', '0.049460ms', '0.032783ms']
```

i) From the table above we can observe that the fastest algorithms are banded and banded R. The reason is that we can save lots of operations by doing the operations within the band of non-zero entries.

Then, the performance of LU and R algorithms are worse than band and band-R, because both of them are operated on the entire matrix, including those zero entries. Specifically, the R algorithm is better than LU algorithm, since the R algorithm is basically doing forward and backword substitution directly, where the LU algorithm needs to compute LU factorization first.

The performance of sparse LU is better then its base version (LU), whereas the performance of sparse R is worse than R, While both of them are slower than band and band R and sparse LU performs much better than sparse R.

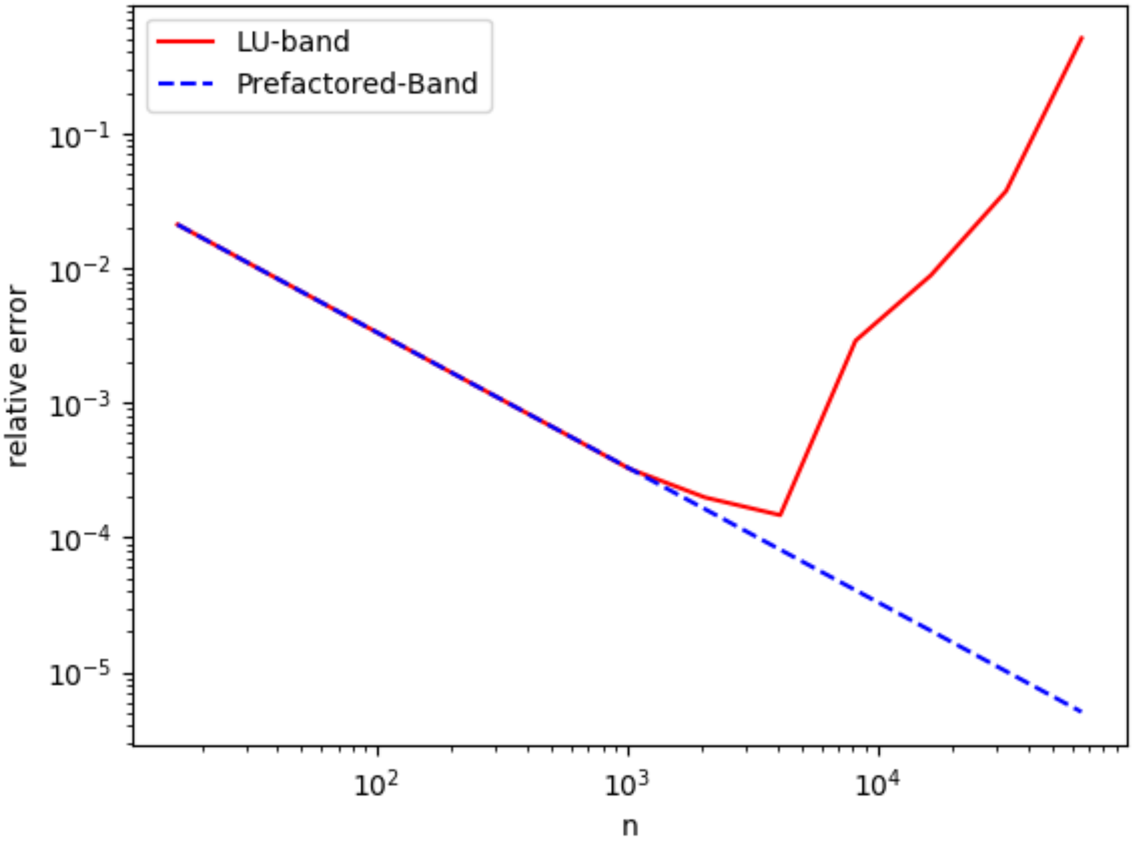
Finally, the performance of Cholesky is only at the middle level, since it needs to compute the cholesky decomposition first, and than solve the linear system.

The order of the performance of these algorithms are roughly:

The performance of R,cholesky,LU algorithms will decrease rapidly when n increases, while the decreasing trend of other algorithms are much Slower, especially for sparse LU and sparseR.

The spsolve_triangular is slower is probably because that this algorithm needs to check regularity and triangularity of A for n times. These additional computations make it slower than spsolve.

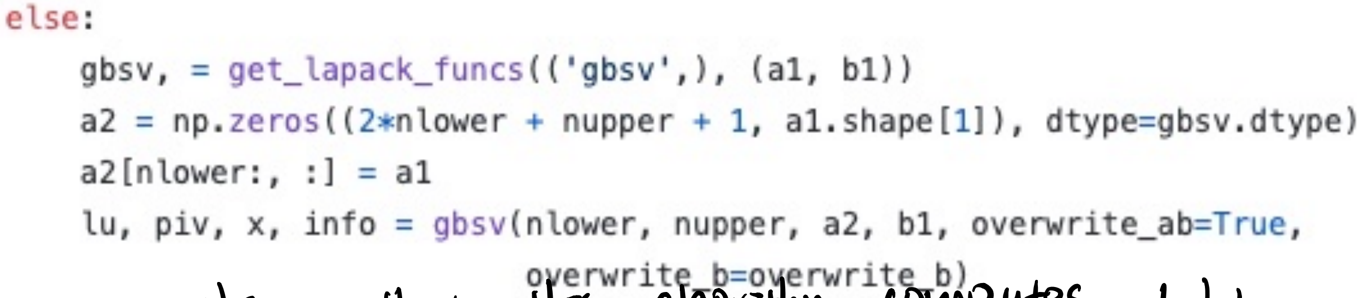
2. a)



b) The accuracy of prefactored banded algorithm goes higher when n increases. The decreasing rate of the relatively error for prefactored banded algorithm is nearly constant, so it looks like a straight line with negative slope on the graph.

The accuracy of LU banded algorithm goes higher when n increases in the early stage. When n gets large enough, the relatively error will go back to the higher level and keep increasing as n increases.

From the source code of sla.solve_banded:



```
else:
    gbsv, = get_lapack_funcs(('gbsv',), (a1, b1))
    a2 = np.zeros((2*nlower + nupper + 1, a1.shape[1]), dtype=gbsv.dtype)
    a2[nlower:, :] = a1
    lu, piv, x, info = gbsv(nlower, nupper, a2, b1, overwrite_ab=True,
                            overwrite_b=overwrite_b)
```

We can observe that this algorithm computes LU decomposition of matrix A, which is not a triangular matrix. According to week 6's lecture, we know that doing LU computations on computer system will result errors, although the error might be very small. However in prefactored banded algorithm, a is decomposed manually already, and the decomposed matrixes R and R^T are all triangular matrices. Therefore, the error caused by this algorithm should be less that LU decompositing matrix A directly. Therefore, this may explain why prefactored approach is more accurate than the LU approach.