```
	for(j = 0; j < n; j+=nb){

		jb = (( nb < n-j ) ? ( nb ) : ( n-j ));

//		load A[ j:j+jb-1 ][ j:j+jb-1 ] to cache ( note: lower triangular part only )

		for(i = 0; i < j; i++){

//			load A[ j:j+jb-1 ][ i ] to cache

			for ( jj = j ; jj < j+jb ; jj++ ){

				for ( ii = jj ; ii < j+jb ; ii++ ){

					A[ ii ][ jj ] -= A[ ii ][ i ]*A[ jj ][ i ];

				}

			}

//			remove A[ j:j+jb-1 ][ i ] from cache ( note: erase, do not write back )

		}

//		( note: leave A[ j:j+jb-1 ][ j:j+jb-1 ] in cache )

		for ( ii = j ; ii < j+jb ; ii++ ){

			for ( jj = j ; jj < ii ; jj++ ){

				A[ ii ][ ii ] -= A[ ii ][ jj ] * A[ ii ][ jj ];

			}

			for ( jj = j ; jj < ii ; jj++ ){

				for ( kk = ii+1 ; kk < j+jb ; kk++ ){

					A[ kk ][ ii ] -= A[ kk ][ jj ] * A[ ii ][ jj ];
				}

			}

			A[ ii ][ ii ] = SQRT_FUN(A[ ii ][ ii ]);

			for ( jj = ii+1 ; jj < j+jb ; jj++ ){

				A[ jj ][ ii ] /= A[ ii ][ ii ];

			}

		}

//		write A[ j:j+jb-1 ][ j:j+jb-1 ] back from cache ( note: lower triangular part only )

		for(i = j+jb; i < n; i+=jb){

			ib = (( jb < n-i ) ? ( jb ) : ( n-i ));

//			load A[ i:i+ib-1 ][ j:j+jb-1 ] to cache

			for(k = 0; k < j; k++){

//				load A[ i:i+ib-1 ][ k ] and A[ j:j+jb-1 ][ k ] to cache

				for(jj = j; jj < j+jb; jj++){

					for(ii = i; ii < i+ib; ii++){

						A[ ii ][ jj ] -= A[ ii ][ k ] * A[ jj ][ k ];

					}

				}

//				remove A[ i:i+ib-1 ][ k ] and A[ j:j+jb-1 ][ k ] from cache
//				( note: erase, do not write back )

			}

//			write A[ i:i+ib-1 ][ j:j+jb-1 ] back from cache

		}

//		load A[ j:j+jb-1 ][ j:j+jb-1 ] in cache ( note: lower triangular part only )

		for(i = j+jb; i < n; i++){

//			load A[ i ][ j:j+jb-1 ] to cache

			for(ii = j; ii < j+jb; ii++){

				for(jj = j; jj < ii; jj++){

					A[ i ][ ii ] -= A[ ii ][ jj ] * A[ i ][ jj ];

				}

				A[ i ][ ii ] /= A[ ii ][ ii ];

			}

//			write A[ i ][ j:j+jb-1 ] back from cache

		}

//		remove A[ j:j+jb-1 ][ j:j+jb-1 ] from cache ( note: erase, do not write back )

	}
```