GPU matrix-vector multiplication

Exercise 4:

1. **OpenMP offload matvec:** Write an OpenMP offload implementation of matrix-vector multiplication on the GPU - you can use your code from the matrix-vector exercise from last week as a starting point.

   Compare the observed runtime to your fastest standard OpenMP version. Did you get any speed-up? Find a way to measure how much time is used for data transfers in your implementations out of the total runtime for different matrix sizes - is it a significant part of the runtime?

   Note: Matrix-vector multiplication is a perfect example of an operation that is **not** efficient to offload to a GPU, when the input matrix is assumed to be located on the host. However, there are many scientific algorithms, where a large number of matrix-vector multiplications is performed using the same input matrix, which is then only transferred to the device once. In the following, we assume such a case.

   Measure the offload execution time without the data transfer for different matrix sizes and once again calculate the speed-up over the fastest standard OpenMP version. Make sure that the speed-up calculation is fair (one CPU vs. one GPU). Is the speed-up as expected?

2. **OpenMP offload matvec - dual GPUs:** Write an OpenMP offload implementation of matrix-vector multiplication that can run simultaneously on two GPUs by splitting the task equally between them (you may assume that the matrix size is an even number). Measure the execution time without transfer and find the speed-up over the single GPU version.

3. **CUBLAS:** Write a function for matrix-vector multiplication that calls the `DGEMV` routine for GPUs from the CUBLAS library. Note that array parameters passed to the routine must live in device memory. The first time you create and destroy a `cublasHandle_t` it has a wake-up time. You timings should be without wake-up.

4. **[Extra] OpenMP offload asynchronous matvec:** Overlapping data transfer with computation can be an effective strategy to increase performance in many offload implementations. Execution of asynchronous data transfers and their synchronization with the target region execution can be specified manually, using the OpenMP clauses: `depend` and `nowait`. The `nowait` clause states that the specified OpenMP task can be run asynchronously with other tasks, thus allowing the data transfer to occur while a target region is executing.

   Write an updated implementation of matrix-vector multiplication that uses asynchronous data transfers and asynchronous execution of target regions in order to overlap the data transfer and computation during execution. To do this the matrix-vector multiplication should be split into several independent sub-calculations: E.g., consider the matrix as divided into slabs, and perform independent matrix-vector multiplications with each of them, to produce separate parts of the result vector.

   Use profiler tool `nsys` to produce a timeline and zoom in to check whether the data transfers and computation are overlapping.