Monthly Summary: October, 2020

Development of the GPU/PSCAD interface continued this month with the variables and methods of the QRFactor class continuing to be refined. In particular, the method of creating the full system sparse matrix was improved by removing the need for an intermediate dense matrix during its construction. Instead, the non-zero entries are stored in an Eigen-specific object called a triplet that can be set dynamically as each subsystem is read in. Once all (column-major ordered) subsystems have been read in, existing Eigen methods are called to allocate the minimum amount of memory required for the sparse matrix for the full system. Once the sparse matrix has been created, Eigen generates the pointers required for the Compressed Storage Row-major (CSR) description that is then passed to the GPU where factoring and solving takes place. Forgoing the expensive and time-consuming step of an intermediate, dense-matrix representation of the full system will result in a significant speed up of data preprocessing, particularly for large systems. For the small test system of a 5×4 sparse matrix, the read-in and conversion to CSR format was timed at 135 μs. It remains to be seen how this will scale with matrix size.

Unlike previous versions of `QRFactor`, the QRFactor interface contains a mix of host (CPU) and device (GPU) functions that are compiled separately by `nvcc` and then linked. This process has resulted in some previously unseen behaviours that have required additional research. At this time, the issue has been narrowed down to passing class variables that have been declared on the host to functions that will be executed on the device. For instance, the formation and conversion of the full system's sparse matrix to CSR format results in the private variables `m_csrRowPtrA`, `m_csrColIndA`, and `m_csrValA` containing the relevant information about the non-zero entries in the full system. The next step in the process is the factoring of the system matrix A, which takes place on the device, and is performed by `__host__ __device__ QRFactor::factor()`. However, upon debugging it is clear that `m_csrRowPtrA`, `m_csrColIndA`, and `m_csrValA` have been re-initialized to placeholder values. This is possibly a consequence of separate compilation of host and device code and will need to be resolved by the use of `cudaMemcpyToSymbol` as well as decorating `m_csrRowPtrA`, *etc.*, with `__device__ __managed__` during their declarations. This would signal to the compiler that the variables can be accessed by either the host *or* the device. Implementation of this is ongoing.