Tasksheet 8:

Task 1. Kronecker Products of Matrices.

Implement a code that will compute the Kronecker product of two matrices. Write a parallel version of the code using OpenMP or in Python, use the multiprocessing module. You may need to work through what a Kronecker product is and some restrictions on how big the matrices can be.

Task 2. The Power Method.

Implement a serial version of the Power method for computing the largest eigenvalue. Two versions of the algorithm were presented. The first version involved to matrix-vector products per iteration of the main loop. Start by implementing this version of the algorithm. To test your code, generate a random matrix of significant size - bigger than a 10×10 matrix. It will be ok to use a small matrix while debugging your code.

Task 3. Reorganizing the Algorithm for Efficiency

Implement the second algorithm discussed in class that involved one matrix-vector product per loop iteration. Compare the algorithms from Task 2 and the code in this task. Do this by timing the codes for increasingly large sizes of the matrix being analyzed.

Task 4. OpenMP implementation of the Power Method.

Implement the Power method with OpenMP directives and runtime routines to parallelize the algorithm.

Task 5. Implementation of the Jacobi Iterative Scheme for Linear Systems.

Implement Jacobi Iteration for the solution of linear systems of equations. Implement the form involving the residual term as discussed in class. Test your code on matrices that are at least 100 rows and 100 columns.