This is the supplementary material for "Synchronization in multilayer networks: when good links go bad" by Belykh, Carter, and Jeter.

The goal of this supplementary material is for readers to be able to compute one of the key quantities in determining the synchronization potential of a specific multilayer network. This quantity, b_k^{int} is the sum of the lengths of all paths passing through edge k that contain edges from both layers in a two layer network.

- 1. This supplement presents an example in the file bk_intExample.m which computes b_k^{int} for each edge in the 20-node network of Fig. 4b, but after edge 13 has been replaced with a y edge. For the network, b_k^{int} is then taken to be the maximum value of b_k^{int} for the entire edge list. When one runs bk_intExample.m without making any changes, it displays an edge list for the entire multilayer network (independent of the layers each edge belongs to) as [Source, Target] and the associated b_k^{int} . The maximum b_k^{int}
- 2. b_k^{int} belongs to edge 14-15, and is equal to 284.
- 3. To utilize this supplement to generate b_k^{int} for any two layer network, the reader simply needs to replace the adjacency lists in xGraph.csv and yGraph.csv with adjacency lists for the two layers in the multilayer network they would like to analyze. These adjacency lists are written in the form:

```
source,target
source,target
source,target
...
```

This code computes $b_k^{\rm int}$ by first assigning a label to each edge in the multilayer network that indicates to which layer the edge belongs. Then, the two edge lists are combined to create a single layer network. Then, it runs Djikstra's algorithm on this single layer network to obtain the shortest path between each pair of nodes. Lastly, for each edge, it counts the lengths of all paths that pass through this edge and contain edges of both types. This computes $b_k^{\rm int}$ for each edge in the multilayer network.

- 4. In addition to computing b_k^{int} for each edge in the multilayer network, the supplement includes the function bk_subgraph.m, which computes the traffic load b_k for a single-layer subgraph of the network. The input to this function is the adjacency list of the subgraph and the output is the subgraph adjacency list and b_k for each edge. bk_subgraph_example.m provides an example using this function to compute the subgraph traffic loads for the 20-node network presented in Fig. 4b.
- 5. The supplement also contains the folder "100-node network" with the adjacency lists for the five 100-node two-layer network configurations of Fig. 6.

Each two-layer network topology is given by two files $xGraph_i$ and $yGraph_i$, i=1,...,5.

The corresponding traffic load b_k^{int} shown in Fig. 6 can be calculated for each of these networks as described above.