**Question 1**

Recall our function **CheckConvergence.m** which uses the difference between a message before and after it is updated (called the ``residual'') as a criterion for convergence. While running LBP with our naive message ordering on the network created by **ConstructRandNetwork** with on-diag weight .3 and off-diag weight .7, print out and plot the residuals of the message 19→3,15→40, and 17→2, with the iteration number on the x-axis (you may want to change the range of the y-axis). Do these messages converge at the same rate? Which converges fastest? (Note, it will be easiest do this assessment within **ClusterGraphCalibrate** and use the helper function **MessageDelta** within that file).

All messages converge at the same rate, though this behavior is not guaranteed in LBP.

All messages converge at the same rate, as is always the case in LBP.

Message 15→40 converges faster than the others, which both take significantly longer.

Message 19→3 converges quickly, followed by 17→2 and finally 15→40 Y

**Question 2**

Which of the following are true about the effects the message passing order can have cluster graph calibration?

Different message passing orders can lead to differences in the final full joint distribution given by cluster potentials.

The value of the final marginals in a graph with no loops can depend on the message passing order.

Different orderings can give different values of the final marginals. Y

On the same graph, one ordering may converge while another never does. Y

**Question 3**

Now, consider the toy image network constructed in ConstructToyNetwork.m. Change the values of the on- and off-diagonal weights of the pairwise factors in this network to different values (which can be done by changing the values passed to this function). First try making the the weights on the diagonal much larger than the off-diagonal weights (1 and .2 respectively), then try the opposite where the off-diagonal weights are much larger (.2 and 1), and then finally try the case where the weights are roughly equal (.5 and .5). For each such model, run LBP and exact inference (using your code from PA4). Which of the following occur in this setup? Why? (NOTE: if LBP does not converge within 100,000 iterations it is okay to truncate the run and report on the pseudo-marginals given at that point)

The (.5,.5) case performs well relative to the others and converges quickly. Y

The case of high on diagonal weights with low off diagonal weights has poor convergence because of high variable correlation coupled with short loops in the network, causing positive feedback loops.

All runs instances converge quickly to approximately the correct marginals due to LBP's ability to overcome local maxima.

The case of high on diagonal weights with low off diagonal weights converges quickly as compared to the others because the strong correlation causes us to quickly enter a strong local optima.

**Question 4**

Let’s run an experiment using our Gibbs sampling method. As before, use the toy image network and set the on-diagonal weight of the pairwise factor (in ConstructToyNetwork.m) to be 1.0 and the off-diagonal weight to be 0.1. Now run Gibbs sampling a few times, first initializing the state to be all 1’s and then initializing the state to be all 2’s. What effect does the initial assignment have on the accuracy of Gibbs sampling? Why does this effect occur?

The initial state is not an important factor in our result as Gibbs can make large moves of multiple variables to quickly escape this bad state.

The initial state has a significant impact on the result as, though our chain mixes quickly, it will mix to a distribution far from the actual distribution and close to the initial assignment.

The initial state has a significant impact on the result of our sampling, which makes sense as strong correlation makes mixing time long and we remain close to the initial assignment for a long time.

The initial state has a significant impact on the result of our sampling as Gibbs will never switch variables because the pairwise potentials enforce strong agreement so we are in a local optima.

**Question 5**

Set the on-diagonal weight of our toy image network to 1 and off-diagonal weight to .2. Now visualize multiple runs with each of Gibbs, MHUniform, Swendsen-Wang variant 1, and Swendsen-Wang variant 2 using VisualizeMCMCMarginals.m (see TestToy.m for how to do this). How do the mixing times of these chains compare? How do the final marginals compare to the exact marginals? Why?

Having strong pairwise potentials enforcing agreement is not a problem for any of these sampling methods and all perform equally well -- mixing quickly and ending up close to the final marginals.

Gibbs outperforms the other variants in this instance. Gibbs has some issues with strong pairwise potentials, but is not nearly as bad as MH where blocks end up stuck with the same level so we cannot mix appropriately.

All variants perform poorly in the case of strong pairwise potentials. All algorithms are subject to positive feedback loops with the tight loops in our grid and strong pairwise agreement potentials, preventing appropriate mixing.

The Swendsen-Wang variants outperform the other approaches, with faster mixing and better final marginals. This is likely due to the block-flipping nature of Swendsen-Wang which allows us to flip blocks and quickly mix in environments with strong agreeing potentials.

**Question 6**

Set the on-diagonal weight of our toy image network to .5 and off- diagonal weight to .5. Now visualize multiple runs with each of Gibbs, MHUniform, Swendsen-Wang variant 1, and Swendsen-Wang variant 2 using VisualizeMCMCMarginals.m (see TestToy.m for how to do this). How do the mixing times of these chains compare? How do the final marginals compare to the exact marginals? Why?

Gibbs and MHUniform perform very well and are somewhat better than the Swendsen-Wang variants. This is because the first two variants use local moves so the local marginals remained consistently close the the true marginals, while SW allows big swings over multiple variables that perturb the distribution.

Gibbs performs poorly relative to the other variants -- exhibiting slower mixing time and marginals further from the exact ones. This difference is likely due to the Gibbs strong global dependence that prevents it from acting appropriately unless all variables are relatively well synced to their true marginals.

All variants perform equally well. They all mix quickly and have very low variance throughout their runs -- remaining close to the true marginals. This is because the pairwise marginals do not force us into preferring agreement when we should not.

Swendsen-Wang outperforms the other variants, though all perform relatively well. SW is better because its larger block moves allow for faster mixing and mean it reaches marginal estimates closer to the true marginals faster.

**Question 7**

When creating our proposal distribution for Swendsen-Wang, if you set all the *qi*,*j*'s to zero, what does Swendsen-Wang reduce to?

Switching *qi*,*j* to 0 leaves us without a valid proposal distribution and is not a feasible sampling algorithm.

Switching *qi*,*j* to 0 is equivalent to MH-Uniform.

Switching *qi*,*j* to 0 is equivalent to the first variant of Swendsen-Wang.

Switching *qi*,*j* to 0 is equivalent to a randomized variant of Gibbs sampling where we are allowed to take a random, rather than fixed, order.