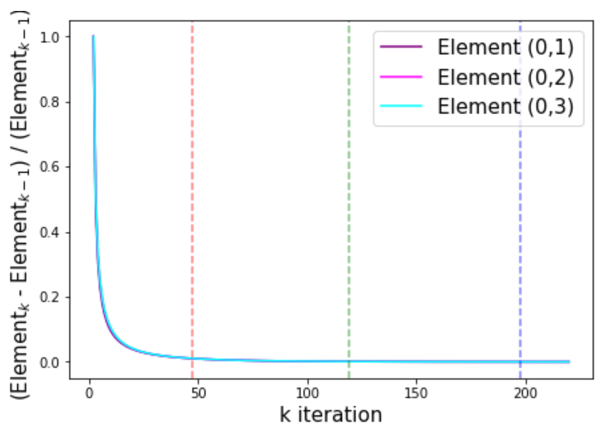
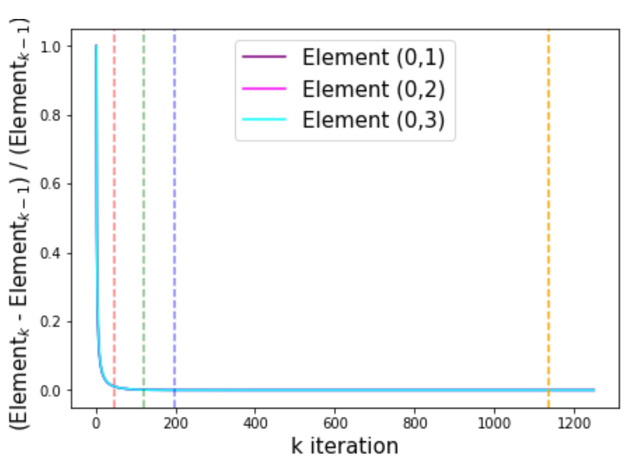
Question 1: Explain the convergence method used for Policy Iteration.

The convergence condition is contingent upon user preference. I settled on tracking the percent difference between consecutive iterations for three different elements. For instance, for the matrix element (0,1) on k=1 I would calculate the percent difference comparing this element with the element (0,1) on k=2. I would then do so for each consecutive pair of iterations (k=3 and k=4, k=4 and k=5, etc.). The convergence parameter to tweak, then, is the desired percent difference.

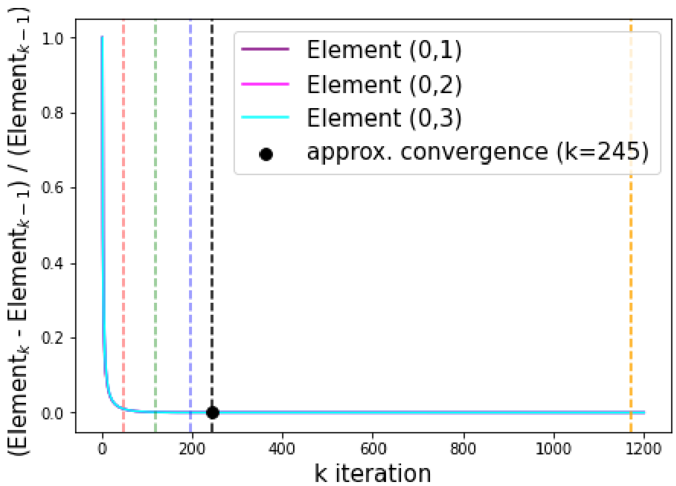
The plot below illustrates a few efforts for three different grid elements and their coordinates. As a start, to approximate convergence, I calculated this percent difference for 100 iterations and determined the lowest k value at which the percentage was 0.01, or 1%, for only element (0,1). This gives the vertical dashed line (red) below, positioned at k=48; for 0.1%, k=120 (dashed, green); and for 0.01%, k=199 (dashed, blue).



Alternatively, if I am looking for the first k value at which two consecutive grid elements (0,2) yield a percent difference of zero, then I will find k=1139 (dashed, orange) to be the desired iteration. This approach, however, does not guarantee the 'actual convergence' k value -- I find that the immediate next iteration that follows this first instance of zero percent difference will itself not necessarily be zero. Moreover, much earlier iterations (e.g., k=700) seem to yield matrices that are only dissimilar to the k=1139 iteration by, say, a one-hundred-thousandth decimal spot. **Note: Despite the aforementioned shortcomings, I use this approach for the .png in the submission folder as well as the printed text.**



I then tried simply rounding the grid's state values each iteration to the third decimal place, simply to evaluate how much sooner this zero percent difference occurred. And indeed, the first instance of 0% is at k=245 for element (0,1). The aforementioned weakness still remains, however, where the next 0% does not appear until k=251.



LASTLY, I found a python function that helps locate the actual convergence point:

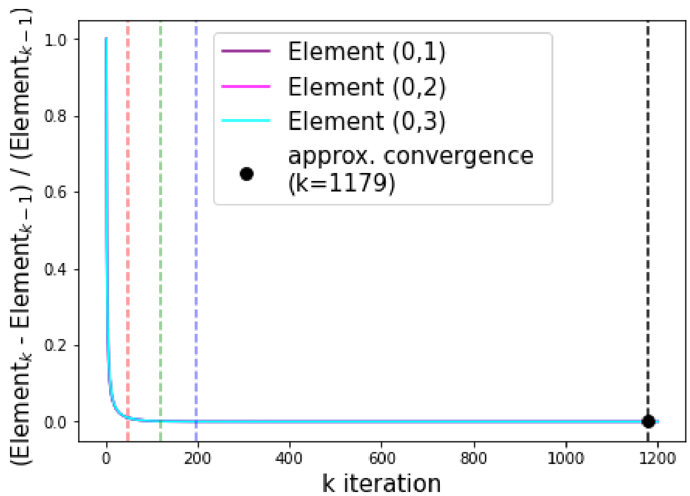
```
In [458]: def ranges(nums):
              nums = sorted(set(nums))
              gaps = [[s, e] for s, e in zip(nums, nums[1:]) if s+1 < e]
              edges = iter(nums[:1] + sum(gaps, []) + nums[-1:])
              return list(zip(edges, edges))

In [434]: a=np.where(eval_dat_all[:,1] == 0)[0]

In [462]: d=ranges(a)
          d

Out[462]: [(1138, 1138),
           (1140, 1140),
           (1147, 1147),
           (1152, 1152),
           (1155, 1155),
           (1157, 1157),
           (1160, 1160),
           (1162, 1162),
           (1164, 1164),
           (1166, 1166),
           (1168, 1168),
           (1170, 1170),
           (1172, 1174),
           (1176, 1176),
           (1178, 1199)]
```

Omitting the esoteric details I am not yet equipped to understand, I can say this screenshot shows variable a, which is an array of indices at which the percent difference is equal to zero. Using this, I can loop through this array and generate lists of the numbers that appear consecutively in the program. If I isolate this last list, then pull the first index in said list, I have effectively found the k value index at and beyond which the policy matrix ceases to change with successive iterations. Granted, one difficulty is having to ensure that the k value is large enough to ensure that this cessation point is realized in the iterations, but otherwise I think the approach is relatively sound. I add a final plot below which marks this k value (k = index+1, or k = 1178+1 = 1179) for element (0,1).



Regardless, this *convergence method* of evaluating percent differences does well to identify either approximate or near-exact convergences, depending on the user's preferred proximity between consecutive policies and rounding conventions. Moreover, since the three selected elements (with different state values) are just about superimposed for large values of k, I assume that concentrating on any one element to measure convergence is sufficient -- again, for approximation purposes. :-)

Question 2: Explain the convergence method used for Value Iteration.

The process for finding a satisfactory convergence condition here is considerably less involved, largely since the cell values will consist of integers without fail. If I simply calculate the difference between two consecutive policy matrices, then isolate the difference for one of the matrix corners opposite that of either of the terminal states, then I can identify rather quickly the first k value at which this difference is zero; and since we are working with integer values, then I need not worry about the decimal-related troubles described above and can be relatively confident that this first instance of a zero difference is, in fact, the actual convergence point. No plotting necessary!