Code Recommendations:

Lines 98 - 100:

Instead of using a for loop here, just use

reducedMatrix = reducedMatrix - eye(size(reducedMatrix))+ diag(u_prev);

General Approach:

- I have read that the factor loadings for a PCA (and thus the factor loadings that we are finding here), should be the eigenvectors scaled by the square root of their respective eigenvalues (source, source). From looking at the current implementation, it looks like we are currently missing that detail multiplication, but im not certain. Easy fix:

  eigenVectors = eigenVectors * diag(eigenValues.^2)

- I have read that we need to drop (zero out) loading vectors / eigenvectors whose corresponding eigenvalues are negative. I do not see this being done in the current implementation of the PAF. The sources for this are the same as the previous. Let me know if I am missing this in the code. From what I have read, eigenvectors whose eigenvalues are negative should not be included in the computation of the next communalities, and thus we need to have logic to handle that.

- I believe that the current implementation does not properly compute the communalities from the eigenvectors beyond not dropping the negative eigenvalue columns. Currently the method takes

  u_curr = 1 - sum(eigenVectors.^2)

  As the new communalities. I believe that matlab takes this sum column wise: meaning that we are collecting a vector of the sum squared of the columns instead of a column of the sum squared of the rows. According to source, I think this is the wrong approach.

- Style wise, I don't understand the purpose of having lines 101 - 114 outside of the while loop while the while loop accomplishes exactly what is done in this section

- It seems to me that line 109 should be (ignoring the issue with transposing the eigenVecs):

  u_curr = 1 - sum(eigenVectors.^2)

  Instead of

  u_curr = sum(eigenVectors.^2)

- Lines 111 and 127 calculates an absolute vector, not a number. Should be comm_diff = sum(abs(u_curr-u_prev)). When we compare comm_diff as a vector to our desired accuracy in the while statement:

while comm_diff > 10^-3

This returns a logic array. When a single number breaks our accuracy threshold, then the iteration stops. This means that one of our communalities could be accurate with the rest still having yet to converge with the current implementation