private LinResult computeRREF\_helper (double[][] A, double[] b) {

LinResult result = computeREF\_helper(A, b);

System.out.println("beginning rref steps");

double[][] augmentedMat = new double[A.length][A[0].length+1];

for (int i=0; i<result.ref.length; i++) {

for (int j=0; j<result.ref[0].length; j++) {

augmentedMat[i][j] = result.ref[i][j];

}

}

System.out.println("REF = ");

printArray(augmentedMat);

System.out.println("inverse after ref = ");

printArray(result.Ainv);

int rp = 0;

int cp = 0;

for (int k=0; k<result.rank; k++) {

rp = k;

for (int i=0; i<result.pivotRow.length; i++) {

if (result.pivotRow[i] == k) {

cp = i;

}

}

for (int r=rp-1; r >= 0; r--) {

double scalar = augmentedMat[r][cp];

for (int j=cp; j<augmentedMat[0].length; j++) {

augmentedMat[r][j] = augmentedMat[r][j] - scalar\*augmentedMat[rp][j];

}

for (int j=0; j<result.Ainv[0].length; j++) {

result.Ainv[r][j] = result.Ainv[r][j] - scalar\*result.Ainv[rp][j];

}

}

}

System.out.println("rref = ");

printArray(augmentedMat);

System.out.println("inverse after rref = ");

printArray(result.Ainv);

System.out.println("rank = "+result.rank);

System.out.println("number of rows = "+A.length);

if (result.rank != A.length) {

result.Ainv = null;

}

result.rref = augmentedMat;

return result;

}