Homework 2

Due date: September 23, 2015, 11:55PM. Since multiple submissions are allowed in Sakai, submit after completing some part of the homework to avoid last minute time crunch, and/or computer failure problems.

Bibliography: Course lecture notes Lessons 9-12. Textbook pp. 45-54 from Systems chapter, pp. 148–160 from Vectors chapter, pp. 193-204 from Matrices chapter.

Save this file as Homework2Solution.tm in your MATH547/homework directory before starting to work on the solution.

1 and 2. (2 course points) Refer to the systems in exercises C21-C28 on page 55. For each system use Octave to determine the rank of the system matrix, and the dimension of the null space and left null space. Then consider the right hand side vector and state whether the system admits no solution, an unique solution or an infinity of solutions. Verify these conclusions by investigation of the reduced row echelon form of the augmented matrix.

(C21 *example solution*, 0.25 course points awarded for reading). In this exercise we move away from tedious hand computation of row echelon forms or bases for the matrix fundamental subspaces. These tasks are carried out by the computer. We concentrate on understanding the relationships between the four fundamental matrix subspaces, and where the right hand side vector is placed, and use the capability of TeXmacs to intersperse comments and calculations (Note: if you carry out calculations outside of TeXmacs, you must copy and paste results to obtain a single document answer for this exercise).

We have $\boldsymbol{A} \in \mathbb{R}^{3\times 4}$, 3 equations, 4 unkowns, domain is $\mathbb{R}^4$ with dimension $n=4$, codomain is $\mathbb{R}^3$ with dimension $m=3$

```
octave> A=[1 4 3 -1; 1 -1 1 2; 4 1 6 5]; b=[5 6 9]'; r=rank(A); disp(r); format short;
    2
octave>
```

System has rank $r=2$, dimension of null space is $n-r=2$, dimension of left null space is $m-r=1$. If $\boldsymbol{b}$ is in the left null space, $\boldsymbol{b}\in N(A^T)$, the system has no solution. If $\boldsymbol{b}$ is in the column space, $\boldsymbol{b}\in C(\boldsymbol{A})$ the system has an infinity of solutions. Find an orthonormal basis for the column space, $\boldsymbol{C}=(\boldsymbol{c}_1\ \boldsymbol{c}_2\ )$

```
octave> C=orth(A); disp(C);
      0.30703  -0.90268
      0.20728   0.37263
      0.92885   0.21522
octave>
octave>
```

Subtract from $\boldsymbol{b}$ its components along the null space basis vector $\boldsymbol{r}=\boldsymbol{b}-(\boldsymbol{b}^T\boldsymbol{c}_1)\boldsymbol{c}_1-(\boldsymbol{b}^T\boldsymbol{c}_2)\boldsymbol{c}_2$

```
octave> r=b-(b'*C(:,1))*C(:,1)-(b'*C(:,2))*C(:,1); disp(r);
      1.6847
      3.7619
     -1.0297
octave>
```

If the remainder vector $\boldsymbol{r}$ is null, then $\boldsymbol{b}\in C(\boldsymbol{A})$, and the system would have an infinite number of solutions. In this case $\boldsymbol{r}\neq\boldsymbol{0}$, hence the system has no solutions (inconsistent linear system). Verify by computing the reduced row echelon form of the augmented matrix

```
octave> disp(rref([A b]));
```

```
    1.00000   0.00000   1.40000   1.40000   0.00000
    0.00000   1.00000   0.40000  -0.60000   0.00000
    0.00000   0.00000   0.00000   0.00000   1.00000
octave>
```

Indeed the last equation after applying row reduction operations would be $0=1$, a false statement, hence the system is not consistent. Notice that instead of computing an orthonormal basis for the column space, we could have computed an orthonormal basis for the left null space.

```
octave> L=null(A'); disp(L);
    0.30151
    0.90453
   -0.30151
octave>
```

If $\boldsymbol{b}$ has a non-zero component in the left null space, the system has no solution. Indeed computation of $\boldsymbol{b}^T\boldsymbol{l}_1$ gives a non-zero scalar, confirming above conclusions.

```
octave> disp(b'*L(:,1));
   4.2212
octave>
```

Apply the above procedure to all the remaining systems (C22-C28). You must include text commenting your results. Just presenting the results of the Octave calculations is not sufficient. A shortened form of the above discussion that you can use as a template for (C22-C28) is:

Solution template.

$\boldsymbol{A}\in\mathbb{R}^{3\times 4}$, $m=3, n=4$

```
octave> A=[1 4 3 -1; 1 -1 1 2; 4 1 6 5]; b=[5 6 9]'; r=rank(A); disp(r);
   2
octave>
```

From above $r=\text{rank}(\boldsymbol{A})=2$, left null space dimension is $m-r=1$, null space dimension is $n-r=2$. Let $\boldsymbol{C}\in\mathbb{R}^{3\times 2}$, be a matrix of orthonomal basis vectors for the column space of $\boldsymbol{A}$

```
octave> C=orth(A);
octave>
```

Compute what remains of $\boldsymbol{b}$ after subtracting components along $\boldsymbol{c}_1, \boldsymbol{c}_2$, $\boldsymbol{r}=\boldsymbol{b}-(\boldsymbol{b}^T\boldsymbol{c}_1)\boldsymbol{c}_1-(\boldsymbol{b}^T\boldsymbol{c}_2)\boldsymbol{c}_2$

```
octave> r=b-(b'*C(:,1))*C(:,1)-(b'*C(:,2))*C(:,2); disp(norm(r));
   4.2212
octave>
```

The result is not the null vector, hence $\boldsymbol{b}$ is not in the column space, $\boldsymbol{b}\notin C(\boldsymbol{A})$, and system has no solution. Verify using reduced row echelon form of the augmented matrix

```
octave> disp(rref([A b]));
    1.00000   0.00000   1.40000   1.40000   0.00000
    0.00000   1.00000   0.40000  -0.60000   0.00000
    0.00000   0.00000   0.00000   0.00000   1.00000
octave>
```

Above is an inconsistent system ($0\cdot x_4=1$ cannot be true).

Remember, what is being verified is your *understanding* of course concepts, not the capacity to carry out Octave commands, hence including appropriate comments as above is required. To be clear, if your homework solution would be just the calculations and a final conclusion, as shown below, no credit would be given at all, even though the calculations and/or conclusion might be correct.

Example of answer that is not awarded any credit.

```
octave> A=[1 4 3 -1; 1 -1 1 2; 4 1 6 5]; b=[5 6 9]'; r=rank(A); disp(r);
    2
octave> C=orth(A);
octave> r=b-(b'*C(:,1))*C(:,1)-(b'*C(:,2))*C(:,2); disp(norm(r));
    4.2212
octave> disp(rref([A b]));
      1.00000   0.00000   1.40000   1.40000   0.00000
      0.00000   1.00000   0.40000  -0.60000   0.00000
      0.00000   0.00000   0.00000   0.00000   1.00000
octave>
```

No solution.

This is also how examination questions will be graded: *calculations without presentation of motivation of why a calculation is being carried out, and analysis of results are not awarded any credit.*

(C22, 0.25 course points). $\boldsymbol{A} \in \mathbb{R}^{3\times 4}$, $m=3, n=4$

```
octave> A=[1 -2 1 -1; 2 -4 1 1; 1 -2 -2 3]; disp(A); b=[3 2 1]'; r=rank(A); disp(r);
      1  -2   1  -1
      2  -4   1   1
      1  -2  -2   3
    3
octave>
```

From above $r = \text{rank}(\boldsymbol{A}) = 3$, left null space dimension is $m - r = 0$, null space dimension is $n - r = 1$. Since $r = 3$, any $\boldsymbol{b} \in \mathbb{R}^3$ is in the column space, $\boldsymbol{b} \in C(\boldsymbol{A})$. Since $n = 4 > 3 = r$, the system has a one-parameter family of solutions, $\boldsymbol{x} = \boldsymbol{x}_p + \lambda \boldsymbol{n}_1$ where $\boldsymbol{x}_p$ is a particular solution of $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, and $\boldsymbol{N} = (\boldsymbol{n}_1) \in \mathbb{R}^{4\times 1}$ is a basis set for the null space. Compute rref($\boldsymbol{A}$)

```
octave> disp(rref([A b]));
      1  -2   0   0   3
      0   0   1   0  -2
      0   0   0   1  -2
octave> null(A)
```

$$\begin{pmatrix} 0.89443 \\ 0.44721 \\ 0 \\ 0 \end{pmatrix}$$

```
octave> [2; 1; 0; 0]/sqrt(5.)
```

$$\begin{pmatrix} 0.89443 \\ 0.44721 \\ 0 \\ 0 \end{pmatrix}$$

```
octave> C=orth(A);
octave> C(:,1)'*C(:,2)
```

$-1.6653e-16$

```
octave>
```

Obtain particular solution $\boldsymbol{x}_p$, null space basis vector $\boldsymbol{n}_1$

$$\boldsymbol{x}_p = \begin{pmatrix} 3 \\ 0 \\ -2 \\ -2 \end{pmatrix}, \boldsymbol{n}_1 = \begin{pmatrix} 2 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

(C23, 0.25 course points). $\boldsymbol{A} \in \mathbb{R}^{3\times 4}$, $m = 3, n = 4$

```
octave> A=[1 -2 1 -1; 1 1 1 -1; 1 0 1 -1]; disp(A); b=[3 1 2]'; r=rank(A); disp(r);
      1  -2   1  -1
      1   1   1  -1
      1   0   1  -1
    2
octave>
```

From above $r = \text{rank}(\boldsymbol{A}) = 2$, left null space dimension is $m - r = 1$, null space dimension is $n - r = 2$. Compute basis $\boldsymbol{C} = (\boldsymbol{c}_1\, \boldsymbol{c}_2)$ for $C(\boldsymbol{A})$ and check if $\boldsymbol{b}$ is in column space by computing $\boldsymbol{r} = \boldsymbol{b} - (\boldsymbol{b}^T\boldsymbol{c}_1)\boldsymbol{c}_1 - (\boldsymbol{b}^T\boldsymbol{c}_2)\boldsymbol{c}_2$

```
octave> C=orth(A); disp(C);
      0.74751  -0.60811
      0.41007   0.73900
      0.52255   0.28997
octave> r=b-(b'*C(:,1))*C(:,1)-(b'*C(:,2))*C(:,2); disp(norm(r));
    0.26726
octave>
```

The remainder is not in the column space. System has no solution. Use rref to verify

```
octave> disp(rref([A b]));
      1   0   1  -1   0
      0   1   0   0   0
      0   0   0   0   1
octave>
```

Indeed, last equation is inconsistent.

(C24, 0.25 course points). $\boldsymbol{A} \in \mathbb{R}^{3\times 4}$, $m = 3, n = 4$

```
octave> A=[1 -2 1 -1; 1 1 1 -1; 1 0 1 -1]; disp(A); b=[2 2 2]'; r=rank(A); disp(r);
      1  -2   1  -1
      1   1   1  -1
      1   0   1  -1
    2
octave>
```

From above $r = \text{rank}(\boldsymbol{A}) = 2$, left null space dimension is $m - r = 1$, null space dimension is $n - r = 2$. Compute basis $\boldsymbol{C} = (\boldsymbol{c}_1\, \boldsymbol{c}_2)$ for $C(\boldsymbol{A})$ and check if $\boldsymbol{b}$ is in column space by computing $\boldsymbol{r} = \boldsymbol{b} - (\boldsymbol{b}^T\boldsymbol{c}_1)\boldsymbol{c}_1 - (\boldsymbol{b}^T\boldsymbol{c}_2)\boldsymbol{c}_2$

```
octave> C=orth(A); disp(C);
      0.74751  -0.60811
      0.41007   0.73900
      0.52255   0.28997
octave> r=b-(b'*C(:,1))*C(:,1)-(b'*C(:,2))*C(:,2); disp(norm(r));
      1.8150e-15
octave>
```

The remainder is null, hence $\boldsymbol{b} \in C(\boldsymbol{A})$. System has a two-parameter family of solutions $\boldsymbol{x} = \boldsymbol{x}_p + \lambda_1 \boldsymbol{n}_1 + \lambda_2 \boldsymbol{n}_2$, with null space basis $\boldsymbol{N} = (\boldsymbol{n}_1\, \boldsymbol{n}_2) \in \mathbb{R}^{4\times 2}$. Use rref to verify

```
octave> disp(rref([A b]));
      1   0   1  -1   2
      0   1   0   0   0
      0   0   0   0   0
octave> null(A)
```

$$\begin{pmatrix} -0.8165 & 0 \\ 0 & 0 \\ 0.40825 & 0.70711 \\ -0.40825 & 0.70711 \end{pmatrix}$$

```
octave> xp=[2;0;0;0]; z1=[-1;0;1;0]; z2=[1;0;0;1];
octave> A*xp-b
```

$$\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

```
octave> A*(xp+2*z1+3*z2)-b
```

$$\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

```
octave>
```

Obtain

$$\boldsymbol{x}_p = \begin{pmatrix} 2 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \boldsymbol{n}_1 = \begin{pmatrix} -1 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \boldsymbol{n}_2 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

(C25, 0.25 course points). $\boldsymbol{A} \in \mathbb{R}^{4\times 3}$, $m = 4, n = 3$

```
octave> A=[1 2 3; 2 -1 1; 3 1 1; 0 1 2]; disp(A); b=[1 2 4 6]'; r=rank(A); disp(r);
      1   2   3
      2  -1   1
      3   1   1
      0   1   2
    3
octave>
```

From above $r = \text{rank}(\boldsymbol{A}) = 3$, left null space dimension is $m - r = 1$, null space dimension is $n - r = 0$. Compute basis $\boldsymbol{C} = (\boldsymbol{c}_1\, \boldsymbol{c}_2\, \boldsymbol{c}_3)$ for $C(\boldsymbol{A})$ and check if $\boldsymbol{b}$ is in column space by computing $\boldsymbol{r} = \boldsymbol{b} - (\boldsymbol{b}^T\boldsymbol{c}_1)\boldsymbol{c}_1 - (\boldsymbol{b}^T\boldsymbol{c}_2)\boldsymbol{c}_2 \boldsymbol{r} = \boldsymbol{b} - (\boldsymbol{b}^T\boldsymbol{c}_1)\boldsymbol{c}_1 - (\boldsymbol{b}^T\boldsymbol{c}_3)\boldsymbol{c}_3$

```
octave> C=orth(A); disp(C);
      0.6893281   0.4496047  -0.0039686
      0.2903676  -0.5823131  -0.7522841
      0.5589403  -0.5255652   0.5871119
      0.3579095   0.4272567  -0.2989188
octave> r=b-(b'*C(:,1))*C(:,1)-(b'*C(:,2))*C(:,2)-(b'*C(:,3))*C(:,3); disp(norm(r));
    4.9058
octave>
```

The remainder is not in the column space. System has no solution. Use rref to verify

```
octave> disp(rref([A b]));
      1   0   0   0
      0   1   0   0
      0   0   1   0
      0   0   0   1
octave>
```

Indeed, last equation is inconsistent.

(C26, 0.25 course points). $\boldsymbol{A} \in \mathbb{R}^{4\times 3}$, $m=4, n=3$

```
octave> A=[1 2 3; 2 -1 1; 3 1 1; 0 5 2]; disp(A); b=[1 2 4 1]'; r=rank(A); disp(r);
      1   2   3
      2  -1   1
      3   1   1
      0   5   2
    3
octave>
```

From above $r = \text{rank}(\boldsymbol{A}) = 3$, left null space dimension is $m - r = 1$, null space dimension is $n - r = 0$. Compute basis $\boldsymbol{C} = (\boldsymbol{c}_1\, \boldsymbol{c}_2\, \boldsymbol{c}_3)$ for $C(\boldsymbol{A})$ and check if $\boldsymbol{b}$ is in column space by computing $\boldsymbol{r} = \boldsymbol{b} - (\boldsymbol{b}^T\boldsymbol{c}_1)\boldsymbol{c}_1 - (\boldsymbol{b}^T\boldsymbol{c}_2)\boldsymbol{c}_2\boldsymbol{r} - (\boldsymbol{b}^T\boldsymbol{c}_3)\boldsymbol{c}_3$

```
octave> C=orth(A); disp(C);
      0.532010  -0.208514  -0.728443
      0.030972  -0.625543  -0.190546
      0.315205  -0.625543   0.605379
      0.785272   0.417029   0.258028
octave> r=b-(b'*C(:,1))*C(:,1)-(b'*C(:,2))*C(:,2)-(b'*C(:,3))*C(:,3); disp(norm(r));
      2.8657e-15
octave>
```

The remainder is null, hence $\boldsymbol{b} \in C(\boldsymbol{A})$. Since the null space is of dimension zero, the system has a unique solution. Use rref to verify

```
octave> disp(rref([A b]));
      1.00000   0.00000   0.00000   1.33333
      0.00000   1.00000   0.00000   0.33333
      0.00000   0.00000   1.00000  -0.33333
      0.00000   0.00000   0.00000   0.00000
octave>
```

The solution is

$$\boldsymbol{x} = \begin{pmatrix} 4/3 \\ 1/3 \\ -1/3 \end{pmatrix}$$

(C27, 0.25 course points). $\boldsymbol{A} \in \mathbb{R}^{4\times 3}$, $m=4, n=3$

```
octave> A=[1 2 3; 2 -1 1; 1 -8 -7; 0 1 1]; disp(A); b=[0 2 1 0]'; r=rank(A); disp(r);
      1   2   3
      2  -1   1
      1  -8  -7
      0   1   1
    2
```

```
octave>
```

From above $r = \text{rank}(\boldsymbol{A}) = 2$, left null space dimension is $m - r = 2$, null space dimension is $n - r = 1$. Compute basis $\boldsymbol{C} = (\boldsymbol{c}_1\ \boldsymbol{c}_2\ )$ for $C(\boldsymbol{A})$ and check if $\boldsymbol{b}$ is in column space by computing $\boldsymbol{r} = \boldsymbol{b} - (\boldsymbol{b}^T\boldsymbol{c}_1)\boldsymbol{c}_1 - (\boldsymbol{b}^T\boldsymbol{c}_2)\boldsymbol{c}_2\boldsymbol{r}$

```
octave> C=orth(A); disp(C);
    -0.304226   0.511252
     0.015786   0.844532
     0.944251   0.155307
    -0.124848   0.035595
octave> r=b-(b'*C(:,1))*C(:,1)-(b'*C(:,2))*C(:,2); disp(norm(r));
   0.80378
octave>
```

The result is not the null vector, hence $\boldsymbol{b}$ is not in the column space, $\boldsymbol{b} \notin C(\boldsymbol{A})$, and system has no solution. Verify using reduced row echelon form of the augmented matrix

```
octave> disp(rref([A b]));
     1   0   1   0
     0   1   1   0
     0   0   0   1
     0   0   0   0
octave>
```

Above is an inconsistent system ($0 \cdot x_3 = 1$ cannot be true).

(C28, 0.25 course points). $\boldsymbol{A} \in \mathbb{R}^{4\times 3}$, $m = 4, n = 3$

```
octave> A=[1 2 3; 2 -1 1; 1 -8 -7; 0 1 1]; disp(A); b=[1 2 1 0]'; r=rank(A); disp(r);
     1   2   3
     2  -1   1
     1  -8  -7
     0   1   1
    2
octave>
```

From above $r = \text{rank}(\boldsymbol{A}) = 2$, left null space dimension is $m - r = 2$, null space dimension is $n - r = 1$. Compute basis $\boldsymbol{C} = (\boldsymbol{c}_1\ \boldsymbol{c}_2\ )$ for $C(\boldsymbol{A})$ and check if $\boldsymbol{b}$ is in column space by computing $\boldsymbol{r} = \boldsymbol{b} - (\boldsymbol{b}^T\boldsymbol{c}_1)\boldsymbol{c}_1 - (\boldsymbol{b}^T\boldsymbol{c}_2)\boldsymbol{c}_2\boldsymbol{r}$

```
octave> C=orth(A); disp(C);
    -0.304226   0.511252
     0.015786   0.844532
     0.944251   0.155307
    -0.124848   0.035595
octave> r=b-(b'*C(:,1))*C(:,1)-(b'*C(:,2))*C(:,2); disp(norm(r));
     9.9959e-16
octave>
```

The remainder is null, hence $\boldsymbol{b} \in C(\boldsymbol{A})$. System has one-parameter family of solutions $\boldsymbol{x} = \boldsymbol{x}_p + \lambda_1\boldsymbol{n}_1$, with null space basis $\boldsymbol{N} = (\boldsymbol{n}_1) \in \mathbb{R}^{4\times 1}$. Use rref to verify

```
octave> disp(rref([A b]));
     1   0   1   1
     0   1   1  -0
     0   0   0   0
```

```
      0    0    0    0
octave>
```

Obtain

$$\boldsymbol{x}_p = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \boldsymbol{n}_1 = \begin{pmatrix} -1 \\ -1 \\ 1 \end{pmatrix},$$

3. (1 course point) Textbook Reading Questions 1, 2 on page 204

Solution. Apply Gauss-Jordan algorithm to compute inverse. Q1: (all steps shown, hand computation)

$$\begin{pmatrix} -2 & 3 & 1 & 0 \\ -3 & 4 & 0 & 1 \end{pmatrix} \sim \begin{pmatrix} 1 & -3/2 & -1/2 & 0 \\ -3 & 4 & 0 & 1 \end{pmatrix} \sim \begin{pmatrix} 1 & -3/2 & -1/2 & 0 \\ 0 & -1/2 & -3/2 & 1 \end{pmatrix} \sim \begin{pmatrix} 1 & -3/2 & -1/2 & 0 \\ 0 & 1 & 3 & -2 \end{pmatrix} \sim$$

$$\begin{pmatrix} 1 & 0 & 4 & -3 \\ 0 & 1 & 3 & -2 \end{pmatrix}$$

$$A^{-1} = \begin{pmatrix} 4 & -3 \\ 3 & -2 \end{pmatrix}$$

Q2: (all steps shown, but with computations done in octave)

```
octave> A=[2 3 1; 1 -2 -3; -2 4 6]; AI=[A eye(3)]; format rat; disp(AI);
           2           3           1           1           0           0
           1          -2          -3           0           1           0
          -2           4           6           0           0           1
octave> AI(1,:)=AI(1,:)/2; disp(AI);
           1         3/2         1/2         1/2           0           0
           1          -2          -3           0           1           0
          -2           4           6           0           0           1
octave> AI(2,:)=AI(2,:)+(-1)*AI(1,:); AI(3,:)=AI(3,:)+2*AI(1,:); disp(AI);
           1         3/2         1/2         1/2           0           0
           0        -7/2        -7/2        -1/2           1           0
           0           7           7           1           0           1
octave> AI(2,:)=(-2/7)*AI(2,:); disp(AI);
           1         3/2         1/2         1/2           0           0
          -0           1           1         1/7        -2/7          -0
           0           7           7           1           0           1
octave> AI(3,:)=AI(3,:)+(-7)*AI(2,:); disp(AI);
           1         3/2         1/2         1/2           0           0
          -0           1           1         1/7        -2/7          -0
           0           0           0           0           2           1
```

Matrix is singular, and does not have an inverse.

4. (1 course point) Compute the inverse of $\boldsymbol{A} \in \mathbb{R}^{m \times m}$ for matrices in exercises C16-C19, p. 205 by row echelon reduction of the full augmented form $(\ \boldsymbol{A}\ \ \boldsymbol{I}\ )$, with $\boldsymbol{I} \in \mathbb{R}^{m \times m}$, the identity matrix. Show intermediate steps. Verify your result by using the Octave `inv(A)` command.

Solution. C16. Perform row reduction and compare final result to inv(A)

```
octave> A=[1 0 1; 1 1 1; 2 -1 1]; AI=[A eye(3)]; disp(AI);
           1           0           1           1           0           0
           1           1           1           0           1           0
           2          -1           1           0           0           1
```

```
octave> AI(2,:) = AI(2,:) - AI(1,:); AI(3,:) = AI(3,:) - 2*AI(1,:); disp(AI);
   1   0   1   1   0   0
   0   1   0  -1   1   0
   0  -1  -1  -2   0   1
octave> AI(3,:) = AI(3,:) + AI(2,:); disp(AI);
   1   0   1   1   0   0
   0   1   0  -1   1   0
   0   0  -1  -3   1   1
octave> AI(3,:) = -AI(3,:); disp(AI);
   1   0   1   1   0   0
   0   1   0  -1   1   0
  -0  -0   1   3  -1  -1
octave> AI(1,:) = AI(1,:) - AI(3,:); disp(AI);
   1   0   0  -2   1   1
   0   1   0  -1   1   0
  -0  -0   1   3  -1  -1
octave> disp(inv(A));
  -2   1   1
  -1   1   0
   3  -1  -1
octave>
```

C17. Perform row reduction and compare final result to inv(A)

```
octave> A=[2 -1 1; 1 2 1; 3 1 2]; AI=[A eye(3)]; disp(AI);
   2  -1   1   1   0   0
   1   2   1   0   1   0
   3   1   2   0   0   1
octave> AI(1,:)=(1/2)*AI(1,:); disp(AI);
   1  -1/2  1/2  1/2   0   0
   1   2    1    0    1   0
   3   1    2    0    0   1
octave>

octave> AI(2,:) = AI(2,:) - AI(1,:); AI(3,:) = AI(3,:) - 3*AI(1,:); disp(AI);
   1  -1/2  1/2  1/2   0   0
   0   5/2  1/2 -1/2   1   0
   0   5/2  1/2 -3/2   0   1
octave> AI(2,:)=2/5*AI(2,:); disp(AI);
   1  -1/2  1/2  1/2   0   0
   0    1   1/5 -1/5  2/5  0
   0   5/2  1/2 -3/2   0   1
octave>

octave> AI(3,:) = AI(3,:) - 5/2*AI(2,:); disp(AI);
   1  -1/2  1/2  1/2   0   0
   0    1   1/5 -1/5  2/5  0
   0    0    0   -1   -1   1
```

```
octave>
```

Matrix is singular, and has no inverse.

C18. Perform row reduction and compare final result to inv(A)

```
octave> A=[1 3 1; 1 2 1; 2 2 1]; AI=[A eye(3)]; disp(AI);
   1   3   1   1   0   0
   1   2   1   0   1   0
   2   2   1   0   0   1
octave> AI(2,:)=AI(2,:)-AI(1,:); AI(3,:)=AI(3,:)-2*AI(1,:); disp(AI);
   1   3   1   1   0   0
   0  -1   0  -1   1   0
   0  -4  -1  -2   0   1
octave> AI(2,:)=-AI(2,:); disp(AI);
   1   3   1   1   0   0
  -0   1  -0   1  -1  -0
   0  -4  -1  -2   0   1
octave> AI(3,:)=AI(3,:)+4*AI(2,:); disp(AI);
   1   3   1   1   0   0
  -0   1  -0   1  -1  -0
   0   0  -1   2  -4   1
octave> AI(3,:)=-AI(3,:); disp(AI);
   1   3   1   1   0   0
  -0   1  -0   1  -1  -0
  -0  -0   1  -2   4  -1
octave> AI(1,:)=AI(1,:)-3*AI(2,:)-AI(3,:); disp(AI);
   1   0   0   0  -1   1
  -0   1  -0   1  -1  -0
  -0  -0   1  -2   4  -1
octave> disp(inv(A));
   0  -1   1
   1  -1   0
  -2   4  -1
octave>
```

C19. Perform row reduction and compare final result to inv(A)

```
octave> A=[1 3 1; 0 2 1; 2 2 1]; AI=[A eye(3)]; disp(AI);
   1   3   1   1   0   0
   0   2   1   0   1   0
   2   2   1   0   0   1
octave> AI(3,:)=AI(3,:)-2*AI(1,:); disp(AI);
   1   3   1   1   0   0
   0   2   1   0   1   0
   0  -4  -1  -2   0   1
octave> AI(2,:)=(1/2)*AI(2,:); disp(AI);
   1   3   1   1   0   0
   0   1 1/2   0 1/2   0
   0  -4  -1  -2   0   1
octave> AI(3,:)=AI(3,:)+4*AI(2,:); disp(AI);
```

```
            1          3          1          1          0          0
            0          1        1/2          0        1/2          0
            0          0          1         -2          2          1
octave> AI(2,:)=AI(2,:)-1/2*AI(3,:); disp(AI);
            1          3          1          1          0          0
            0          1          0          1       -1/2       -1/2
            0          0          1         -2          2          1
octave> AI(1,:)=AI(1,:)-3*AI(2,:)-AI(3,:); disp(AI);
            1          0          0          0       -1/2        1/2
            0          1          0          1       -1/2       -1/2
            0          0          1         -2          2          1
octave> disp(inv(A));
            0       -1/2        1/2
            1       -1/2       -1/2
           -2          2          1
octave>
```

5. (4 course points) Apply your knowledge and understanding of matrix subspaces to the face recognition problem. This will be our first true computer application, and we'll use a few more features of Octave.

Preparation. First, let's learn how to define functions in Octave. Here's a simple function that returns the double of the input value. When writing functions in an Octave session within TeXmacs, use Shift+Enter to get a new line, and use Enter to complete the definition.

```
octave> function d=dbl(x)
          d=2*x;
        endfunction;
octave>
```

Test that the function works

```
octave> dbl(2)
   4
octave>
```

Next, let's learn how to work with strings. Strings are enclosed in quotes, and can be assigned as values to variables. You can join multiple strings with the `strcat` function.

```
octave> astr="a"; bstr="b"; cstr="c";
octave> strcat("a","b","c")
   abc
octave> strcat(astr,bstr,cstr)
   abc
octave>
```

You convert numbers to strings using the `num2str` function.

```
octave> x=2.2; strcat("The answer is x=",num2str(x))
   The answer is x=2.2
octave>
```

You can control the format of how a number is converted to a string using C-language formatting directives.

```
octave> num2str(2,"%2.2d")
   02
octave>
```

Using the above, let us define a function to read a face from the yalefaces database

```
octave> function img=readface(n,type)
          fhead = "/home/student/courses/MATH547/lessons/yalefaces/subject";
          fnr = strcat(num2str(n,"%2.2d"),".");
          fname = strcat(fhead,fnr,type);
          [img,map,alpha] = imread(fname);
        endfunction;
octave>
```

Parse the above to understand how things work. A string is formed for the file name using a header, the number of the image we want, and the type of the image. The image is read from the file and returned. We can find out its size and set the number of height, width pixels (h, w). Look at a portion of the matrix to see that an image is simply an array of gray-value intensities with the value 0 denoting black and the value 255 denoting white. We define a function to reformat an image as a vector with entries between 0 and 1.

```
octave> f01n=readface(1,"normal");
octave> [h,w]=size(f01n);
octave> disp([h w]);
      243   320
octave> f01n(97:103,97:103)
     255  255  186   83  135  153   79
     255  248  111  149  192  104   62
     255  224  164  251  223  102   46
     255  235  202  255  203   93   45
     255  255  255  246  136   58   50
     255  255  255  231  123   52   51
     255  255  255  175   79   50   58
octave> function vec=img2vec(img)
          [h,w]=size(img); m=h*w;
          vec=zeros(m,1);
          vec=double(reshape(img,m,1))/255.;
        endfunction;
octave> v01n=img2vec(f01n);
octave>
```

Now let us form a matrix of the normal face images $\boldsymbol{A} \in \mathbb{R}^{m \times n}$, with $m = hw = 320$, and $n = 15$ subjects. First create a matrix of the appropriate size with zero entries everywhere.

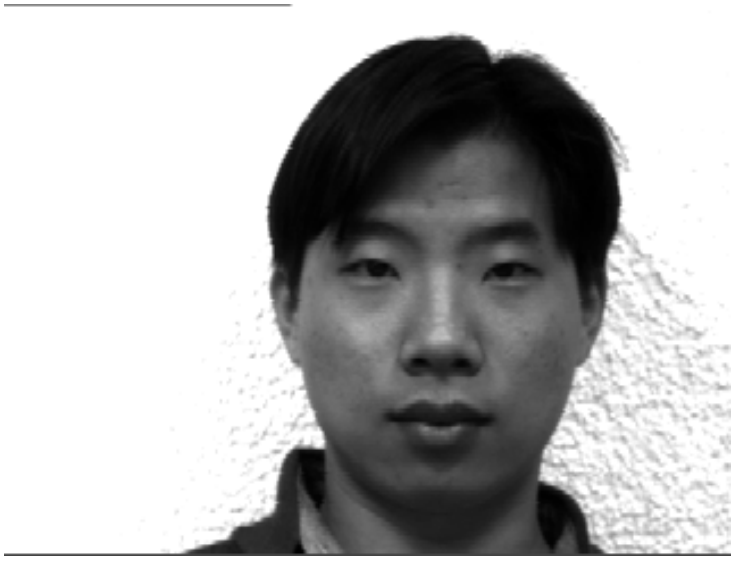
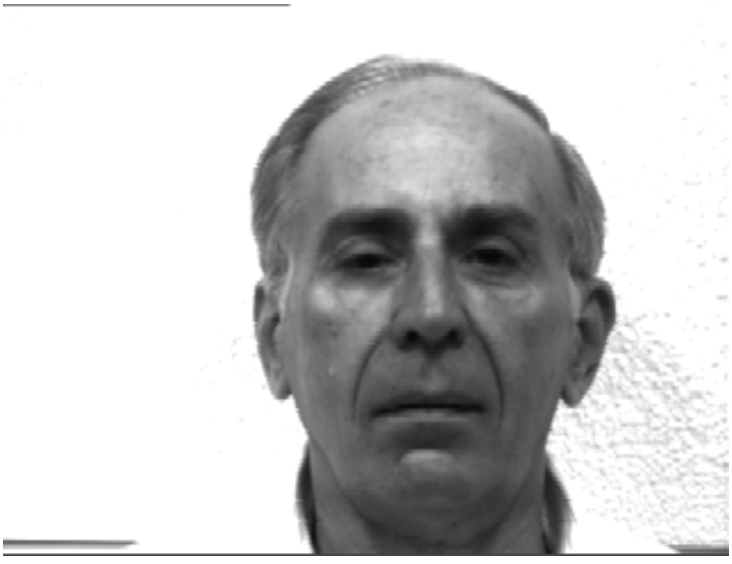
```
octave> m=h*w; n=15; A=zeros(m,n);
octave> rank(A)
   0
octave>
```

Now use a loop to fill each column, with image gray level rescaled to be between 0 and 1.

```
octave> for j=1:n
          fj=readface(j,"normal");
          A(:,j) = img2vec(fj);
        end;
octave>
```

At present there is no command from within Octave to embed an image directly into TeXmacs. This has to done manually by using the menu option Insert->Image->Insert image ... dialogue. Here are the 15 "normal faces"





Since those are 15 rather different people, we should expect the column vectors of the matrix $\boldsymbol{A}$ to be linearly independent as confirmed by computing the rank

```
octave> rank(A)
   15
octave>
```

Let us construct a linear combination of some faces

```
octave> newf=0.5*A(:,1)+0.5*A(:,3);
octave>
```

Define a function that saves the vector to an image file on disk

```
octave> function writeface(f,h,w,name)
          i = reshape(f,h,w);
          fname = strcat("/home/student/courses/MATH547/homework/",name,".png");
          imwrite(i,fname);
        endfunction;
octave> writeface(newf,h,w,"newface");
octave> format short;
octave>
```

The image has been written to a the file newface.png within your homework directory. Here it is (Insert->Small figure followed by Insert->Image->Insert image ...):

Figure 2. Linear combination of subject01 and subject03.

Just from the above you notice there are additional issues that arise:

- image alignment

- feature alignment

These are interesting open research problems. For now start investigating practical application of the concepts learned so far by answering the following.

Questions.

a) Are images from different poses close to the original image? Since two images are vectors $\boldsymbol{u}$, $\boldsymbol{v}$, the smaller the angle

$$\theta = \arccos\left(\frac{\boldsymbol{u}^T \boldsymbol{v}}{\|\boldsymbol{u}\| \, \|\boldsymbol{v}\|} \right)$$

between the vectors, the closer they are. Compute the angle between the various poses and the normal images for at least 5 subjects, present as a table, and comment the result

For example:

```
octave> h01=img2vec(readface(1,"happy"));
octave> n01=img2vec(readface(1,"normal"));
octave> acos(h01'*n01/norm(h01)/norm(n01))
```

0.15307

```
octave> poses=["centerlight"; "glasses"; "happy"; "leftlight"; "noglasses"; "normal";
        "rightlight"; "sad"; "sleepy"; "surprised"; "wink"];
octave> np=max(size(poses)); disp(np);
    11
octave> ns=5; scprod=zeros(np,ns);
        for s=5:4+ns
          nFace = img2vec(readface(s,"normal"));
          for p=1:np
            pFace = img2vec(readface(s,poses(p,:)));
            scprod(p,s-4)=pFace'*nFace/norm(pFace)/norm(nFace);
          end
        end;
octave> disp(poses);
   centerlight
   glasses
   happy
   leftlight
   noglasses
   normal
   rightlight
   sad
   sleepy
   surprised
   wink
octave> disp(scprod);
      0.98753   0.92578   0.97376   0.97685   0.95932
```

```
    0.99739   0.94659   0.98007   1.00000   0.94225
    0.99754   0.99552   0.97788   0.97654   0.99540
    0.91731   0.74995   0.86003   0.84721   0.83625
    0.99681   1.00000   1.00000   0.97203   1.00000
    1.00000   1.00000   1.00000   1.00000   1.00000
    0.85862   0.92884   0.92267   0.90360   0.91918
    0.99757   0.98684   0.98477   0.96135   0.98607
    0.99776   0.97054   0.99829   0.96625   0.98717
    0.99415   0.94757   0.98286   0.94264   0.98755
    0.99880   0.98221   0.98319   0.95706   0.97840

octave>
```

The scalar products are close to one, hence the poses are indeed close to the normal image.

b) Another way to measure closeness is to compute the ratio of the norm of the difference to norm of a vector (this is known as a relative error) $\|\boldsymbol{u}-\boldsymbol{v}\|/\|\boldsymbol{v}\|$. Do this for poses, subjects chosen in (a) and compare results

```
octave> ns=5; relerr=zeros(np,ns);
        for s=5:4+ns
          nFace = img2vec(readface(s,"normal"));
          for p=1:np
            pFace = img2vec(readface(s,poses(p,:)));
            relerr(p,s-4)=norm(pFace-nFace)/norm(nFace);
          end
        end;
octave> format short; disp(relerr);
    0.15745   0.41111   0.22762   0.21566   0.29218
    0.07221   0.32823   0.20040   0.00000   0.33962
    0.07004   0.09477   0.21233   0.21683   0.09600
    0.39980   0.66663   0.51342   0.53429   0.54873
    0.07982   0.00000   0.00000   0.23540   0.00000
    0.00000   0.00000   0.00000   0.00000   0.00000
    0.57061   0.37995   0.40869   0.44494   0.40316
    0.06973   0.16227   0.17551   0.27878   0.16659
    0.06689   0.24445   0.05857   0.26022   0.16002
    0.10797   0.32476   0.18579   0.33893   0.15768
    0.04904   0.18844   0.18463   0.29096   0.20772

octave>
```

As expected the smaller errors arise from small variations in the image, e.g. "noglasses" for people who do not ordinarily wear glasses. When lighting conditions change markedly the relative error increases.

c) We don't expect an arbitrary image to be within the column space of the normal image $C(\boldsymbol{A})$. But how close can we get? Recall that if $\boldsymbol{b}\in\mathbb{R}^m$ is not in the column space of $\boldsymbol{A}$, $\boldsymbol{b}\notin C(\boldsymbol{A})$, the closest linear combination is found by solving the least squares problem

$$\boldsymbol{A}^T(\boldsymbol{b}-\boldsymbol{A}\boldsymbol{x})=0\Rightarrow(\boldsymbol{A}^T\boldsymbol{A})\boldsymbol{x}=\boldsymbol{A}^T\boldsymbol{b}\Leftrightarrow\boldsymbol{N}\boldsymbol{x}=\boldsymbol{c}$$

The vector within the column space closest to $\boldsymbol{b}$ is then $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$. For each of the poses of a subject solve the least squares problem and show the resulting image. Example: Execute the following random number generator to determine the subject you will work with

```
octave> round(rand*15)
   12
octave> b=img2vec(readface(12,"happy"));
octave> N=A'*A; c=A'*b;
octave> x=N\c;
octave> y=A*x;
octave> writeface(y,h,w,"yHappy12");
octave> strcat("y",poses(1,:),"12")
   ycenterlight12
octave>
```

Figure 3. (Top row): linear combination of normal faces closest to subject12.happy, cent
(Bottom row): leftlight, rightlight, sleepy.

Carry out the calculations, and complete the table in Figure 3 for the other poses.

```
octave> for p=1:np
          b = img2vec(readface(s,poses(p,:)));
          c = A'*b; x = N\c; y = A*x;
          writeface(y,h,w,strcat("y",poses(p,:),"12"));
        end;
octave>
```

Results are in Fig. 3. Notice that slight misalignment of subjects and changes in lighting give rise to large changes in "closest' linear combination.'

d) In what matrix subspace should the differences $\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}$ computed above belong to? Check this numerically.

The differences should be in the left null space, and $\boldsymbol{A}^T(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x})$ should be essentially zero.

```
octave> for p=1:np
          b = img2vec(readface(s,poses(p,:)));
          c = A'*b; x = N\c; y = A*x;
          disp(strcat("For p=",num2str(p)," ||A'*(b-A*x)||=",num2str(norm(A'*(b-y)))));
        end;
  For p=1 ||A'*(b-A*x)||=4.3407e-09
  For p=2 ||A'*(b-A*x)||=6.1051e-09
  For p=3 ||A'*(b-A*x)||=3.6986e-09
  For p=4 ||A'*(b-A*x)||=1.4556e-08
  For p=5 ||A'*(b-A*x)||=1.5332e-11
  For p=6 ||A'*(b-A*x)||=1.5332e-11
  For p=7 ||A'*(b-A*x)||=5.1956e-09
  For p=8 ||A'*(b-A*x)||=3.5041e-09
  For p=9 ||A'*(b-A*x)||=3.5384e-09
  For p=10 ||A'*(b-A*x)||=3.5292e-09
  For p=11 ||A'*(b-A*x)||=3.2699e-09
octave>
```