EE 257 Homework 3 Solutions by Ann Chen

Problem 1

Our system can be written as Ax=y as stated in Lab 2 solution, problem 3. If A = U*S*V', then A'*A= V'*S^2*V and A*A'= U'*S^2*U. Therefore, we can eig(A'*A) and eig(A*A') to compute U, S and V.

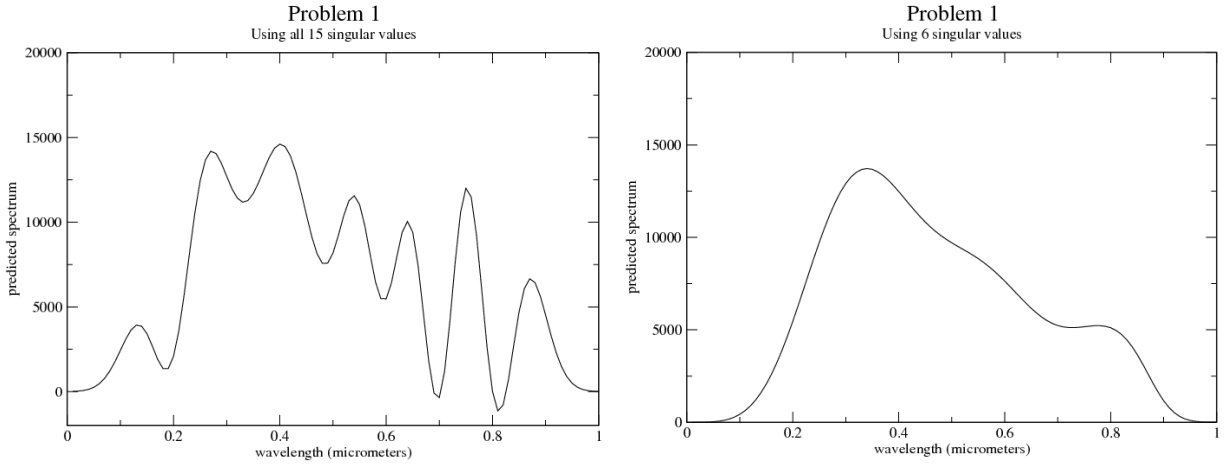

Fig. 1 (Left) the spectrum solution using all 15 singular values of matrix A. (Right) the spectrum solution using the largest 6 singular values of matrix A. The reason to choose 6 largest singular values is stated in the Fig. 2 captions.

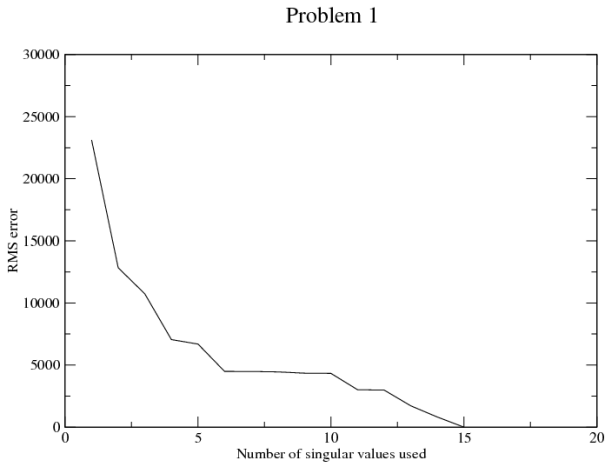

Fig. 2 the number of singular values we use to reconstruct the spectrum vs. RMS error. When we use all 15 singular values, the RMS error is minimized, however, we over-fit the data and the solution is corrupted by noise. If we choose too few singular values, then we may lose too much information to reconstruct the signal. Based on the L curve above, we choose 6 eigenvalues to reconstruct the spectrum solution.

C++ code for problem 1

```
#include <iostream>
#include <fstream>
#include <string>
#include <Eigen/LU>
#include <Eigen/SVD>
#include <Eigen/Dense>

using namespace std;
using namespace Eigen;

int main()
{
  int i,j,k;/*index var*/
  int m=15;
  int n=100;

  /* Read the text input */
  VectorXf y(m);
  ifstream infile ("lab2prob3.dat");
  if (infile.is_open())
  {
    k=0;
    while (k < m)
    {
      infile>>j>>y(k);
      k++;
    }
    infile.close();
  }
  else{
    cout << "Unable to open file";
    return 1;
  }

  /* Construct A*/
  float c;
  VectorXf l(n);
  MatrixXf A(m,n);
  for(i=0; i<m; i++){
    for(j=0;j<n;j++){
      c=0.15+0.05*i;
      l(j)=0.01+0.01*j;
      A(i,j)=exp(-pow((l(j)-c),2)/(2*(0.04*0.04)));
    }
  }
  /* svd(A) */
  JacobiSVD<MatrixXf> svd(A, ComputeFullU | ComputeFullV);
  MatrixXf U = svd.matrixU();
  MatrixXf V = svd.matrixV();
  VectorXf S = svd.singularValues();

  // Use nk largest sigular values to reconstruct the spectrum
  int nk=6;
  MatrixXf SInv = MatrixXf::Zero(n,m);
  for (i = 0; i < nk; i++){
```

```
        SInv(i,i) = 1.0f/S(i);
    }
  VectorXf s = V*SInv*U.transpose()*y;

  /* Write gamma and the rms errors to the output */
  ofstream out;
  out.open ("spectrum_6.txt");
  for(i=0; i < n; i++){
    out << l(i)<<' '<<s(i)<<endl;
  }
  out.close();

  return 0;
}
```

FORTRAN 90 code for problem 1

```
program inversion
  implicit none   ! this statement forces all variables to the defined

  ! Determine the spectrum of a star

  !!declear variables
  character*80 :: fname_in, fname_out1,fname_out2 !input and output filename
  integer :: i,j,m,n,k  ! counter variable
  real,allocatable::y(:),y_pred(:),x(:),A(:,:) ! y=As, where size(A)=15*100
  !A is the inpulse response matrix, y are the measurements of 15 detectors
  !s is the unknown spectrum of the star
  real,allocatable::c(:)! center wavelength at 15 detector
  real,allocatable::l(:)! spectral range
  real,allocatable::U(:,:),Sinv(:,:),S2(:),V(:,:)
  real,allocatable::err(:)

  !! READ COMMAND LINE INPUTS
   if (iargc().lt.3) then
      ! make sure the user inputs a file name, otherwise print proper usage
      print *, 'USAGE: inversion input_file output_rms output_reg'
      print *, '  Determine the spectrum of a star using SVD'
      stop    ! stop if number of inputs is incorrect
   else
      call getarg(1,fname_in)       ! read input file name
      call getarg(2,fname_out1)     ! read output file1 name
      call getarg(3,fname_out2)     ! read output file1 name
   endif

   !!read input data
   m=15;n=100
   allocate(x(n))
   allocate(y(m))
   allocate(A(m,n))
   print *, 'Reading text from file : ',fname_in
   open(21,file=fname_in)  ! open input file, 21 is the file identifier
   do i=1,m
      read(21,*) k,y(i)
   enddo
```

```
close(21)    ! close input file
print *,'all data are read from the input file!'

!!Construct A in y = As
allocate(c(m))
allocate(l(n))
do i=1,m
   c(i)=0.15+0.05*(i-1)!in micrometers
enddo
do j=1,n
   l(j)=0.01+0.01*(j-1)
enddo
!inpulse respose matrix A
do j=1,n
   do i=1,m
      A(i,j)=exp(-(l(j)-c(i))**2/2/0.04**2)
   enddo
 enddo

 !!Singular value decomposition
 allocate(U(m,m))
 allocate(S2(m))
 allocate(Sinv(n,m))
 allocate(V(n,n))
 call eigen(m,matmul(A,transpose(A)),S2,U)
 do i=1,n
    do j=1,m
       if (i==j)then
          Sinv(i,j)=sqrt(1/S2(j))
       else
          Sinv(i,j)=0
       endif
    end do
 end do
 V=transpose(matmul(Sinv,matmul(transpose(U),A)))
 x=matmul(matmul(V,Sinv),matmul(transpose(U),y))

 allocate(y_pred(m))
 allocate(err(m))
 do i=1,15
    x=matmul(matmul(V(:,1:i),Sinv(1:i,1:i)),matmul(transpose(U(:,1:i)),y))
    y_pred=matmul(A,x)
    err(i)=sqrt(sum((y_pred-y)**2)/15)
 enddo

 !write the RMS vs. number of eigen values results to outputfile1
 print *, 'Writing text to ', fname_out1
 open(22,file=fname_out1) ! open output file
 do i = 1,m
    write(22,*) i,' ',err(i)
 enddo
 close(22)
 print *,'RMS calculation: jobs done!'

 k=6
 x=matmul(matmul(V(:,1:k),Sinv(1:k,1:k)),matmul(transpose(U(:,1:k)),y))
 print *, 'Writing text to ', fname_out2
```

```
    open(22,file=fname_out2) ! open output file
    do i = 1,n
       write(22,*) l(i),' ',x(i)
    enddo
    close(22)
    print *,'SVD inversion calculation: jobs done!'

    deallocate(y,y_pred,x,A,c,l,U,Sinv,S2,V,,err)

endprogram inversion
```

Problem 2

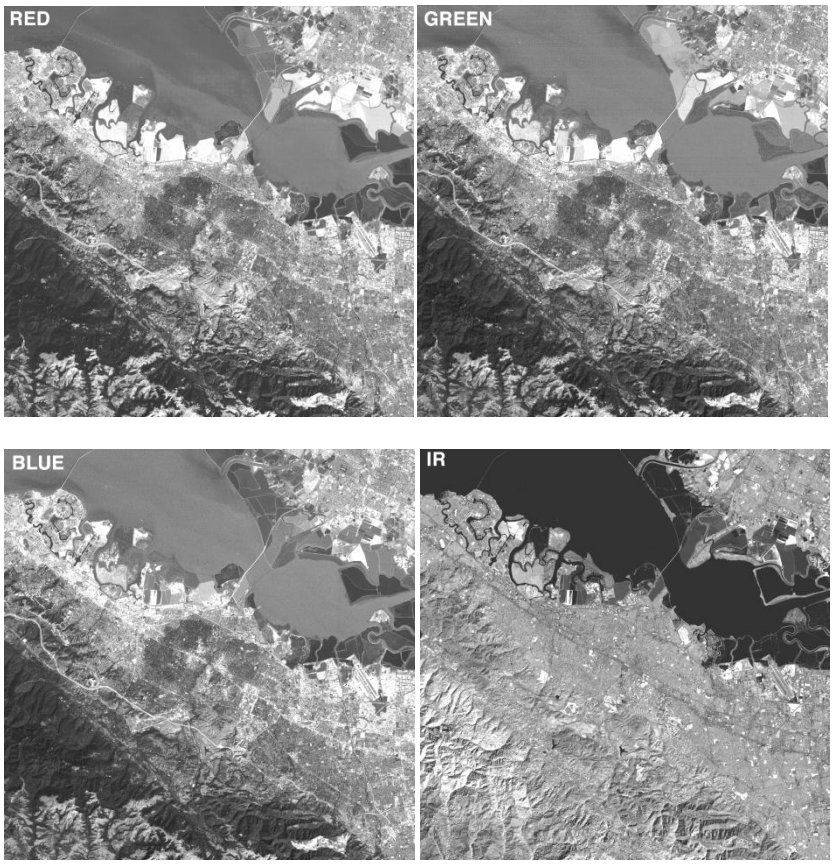

Fig. 3 Scaled red, green, blue and infrared images

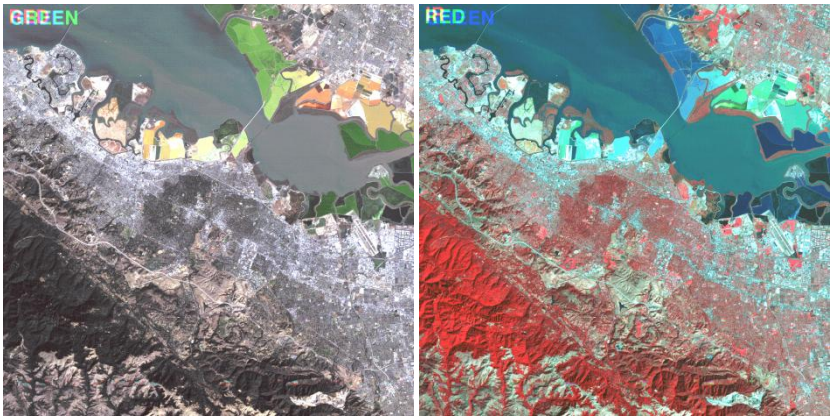

Fig. 4 (Left) natural image (Right) False color image

Problem 3

To generate the image below, please follow the steps in handout #8.

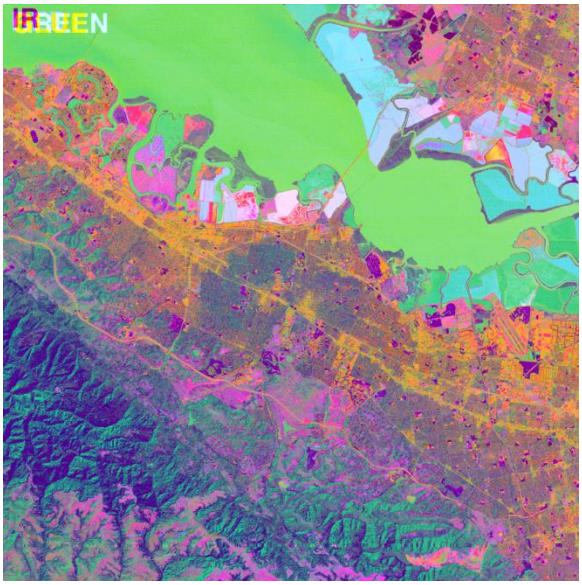

Fig. 5 the PCA image using the largest three PCA components

Note: the source codes for problem 2&3 will be posted in the homework 4 solutions after all homework 4 reports are turned in.