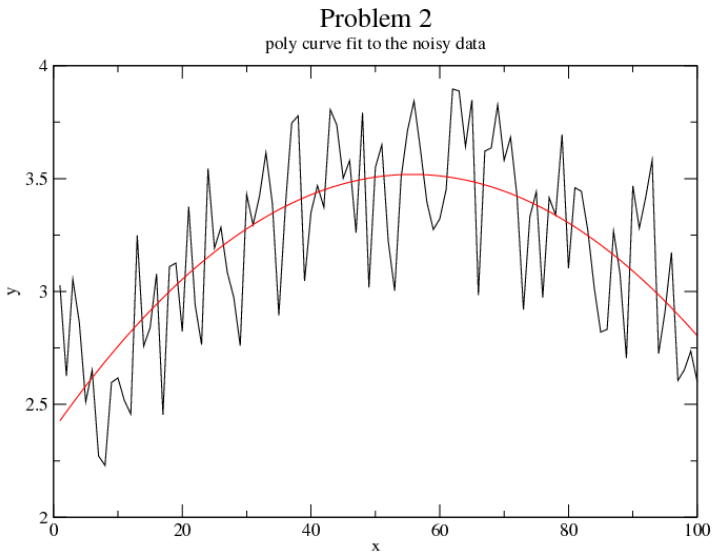
# *EE 257 Homework 2 Solutions by Ann Chen*

## *Problem 1*

```
The results are:
b=a*x:    8.000000        9.000000        5.000000
a*inv(a):           1.000000        0.000000        0.000000
                    0.000000        1.000000        0.000000
                    0.000000        0.000000        1.000000
```

## *Problem 2*




The polynomial coefficients:

C0=2.387476

C1=4.0601373E-02

C2=-3.6432035E-04

y=c0+c1*x+c2*x*2

## *Problem 3 and 4*

We can write the $i^{th}$ detector's data as:

$$y_i = \sum_{j=1}^{100} e^{-\frac{(\lambda_j - center_i)^2}{2*0.04*0.04}} x_j$$

Where $x_j$ is the spectrum we want to measure at $\lambda_j$. Note the unit in the above equation is in micrometers; the formula in the homework problem set is in the unit of meters.

We can rewrite the system as Ax=y, where y(data) is a 15*1 vector, x(model) is a 100*1 vector and A is a 15*100 matrix.

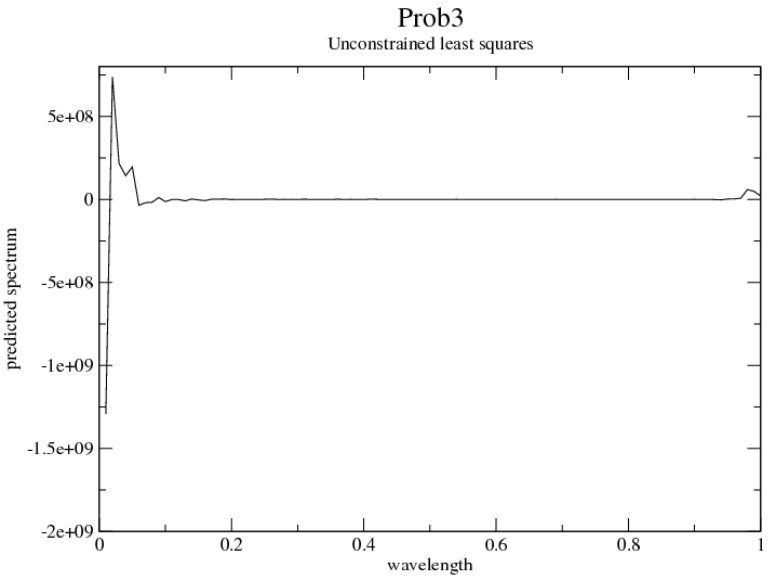
Since A is a fat matrix, the unconstrained least squares solution does not lead to physics sound solution as $A^TA$ is not full rank.

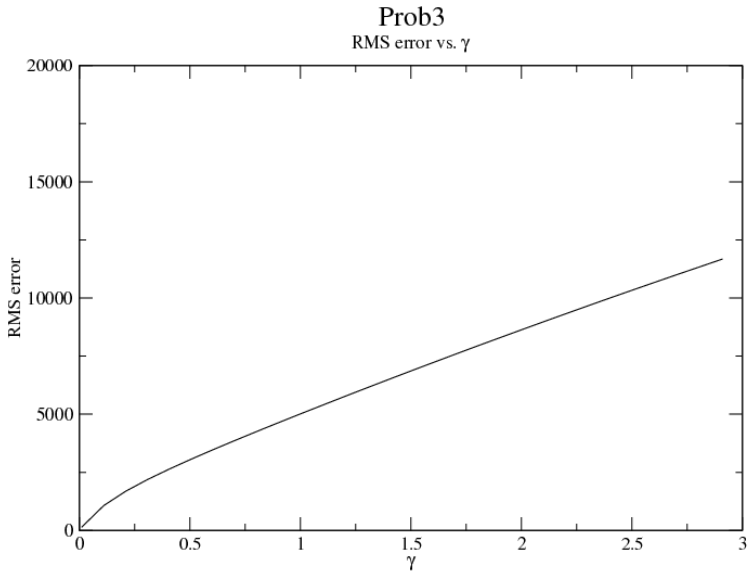
We can get a regularized solution as:

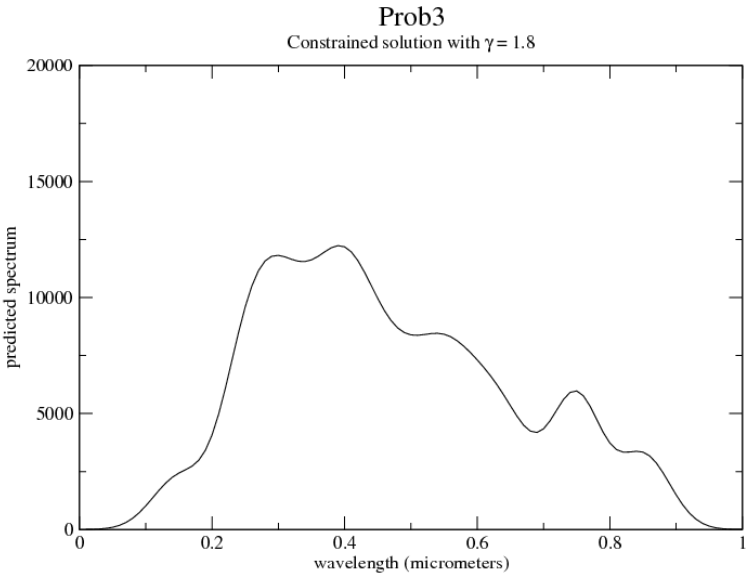
$$x = (A^TA + \gamma H^TH)^{-1}A^Ty$$

H is the identity matrix for problem 3 and first order derivative matrix for problem 4.

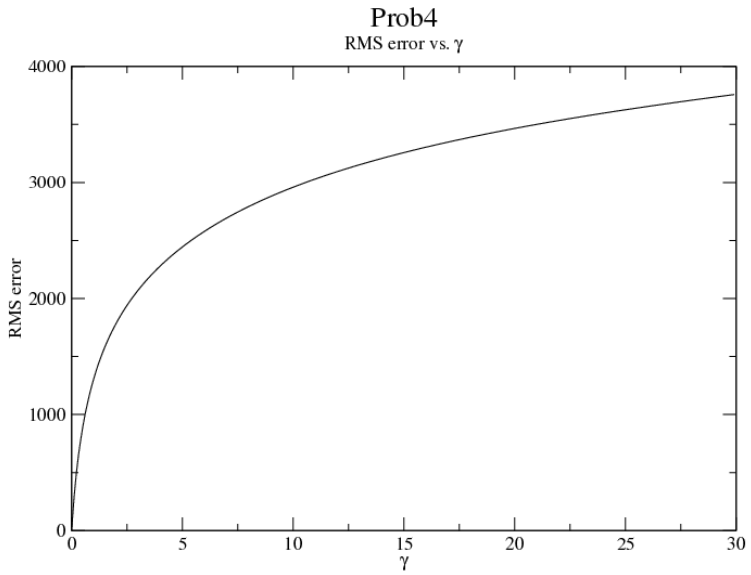
This solution minimizes $\|Ax - y\|^2 + \gamma\|Hy\|^2$. By varying $\gamma$, you make a tradeoff between data fitting error and power/smoothness of the solution.

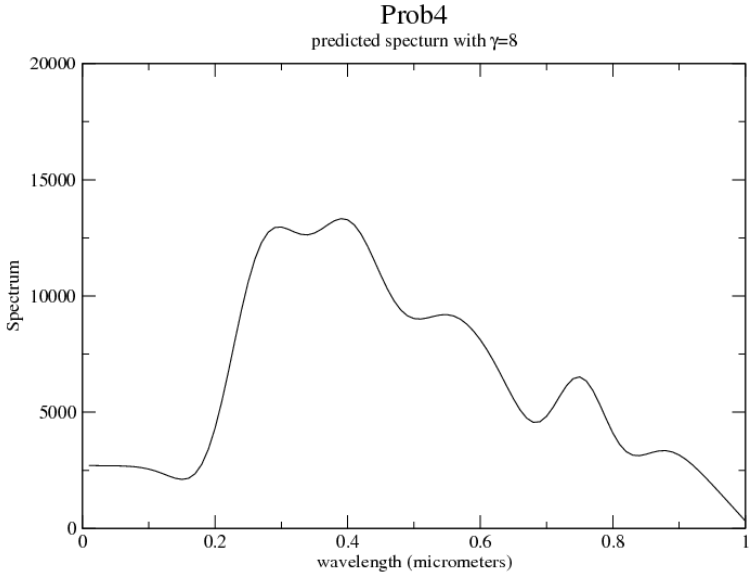





Note. You can increase $\gamma$ and then the two peaks near the 0.4 micrometers -> one peak.




Note. You can increase $\gamma$ and then the two peaks near the 0.4 micrometers -> one peak.

*Problem 3 C++ Source code*

```
#include <iostream>
#include <fstream>
#include <string>
#include <Eigen/LU>
#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)));
    }
  }
  /* Constrained LS */
  MatrixXf Iden = MatrixXf::Identity(100,100);
  int num=100;
  VectorXf gamma(num),err(num);
  for(i=0; i < num; i++){
    gamma(i)=0.01+i*0.1;
    VectorXf s = (A.transpose()*A + gamma(i)*Iden).inverse()*A.transpose()*y;
    VectorXf dy = A*s-y;
    float rms2= dy.dot(dy)/float(m);
    err(i) = sqrt(rms2);
```

```
  }

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

  return 0;
}
```

*Problem 3 FORTRAN Source code*

```
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,fname_out3 !input and
output filename
  integer :: i,j,m,n,k  ! counter variable
  real,allocatable::y(:),s(:),A(:,:),y_pred(:) ! 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::invA(:,:),H(:,:)
  real,allocatable::gamma(:),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_ls output_rms output_reg'
      print *, '  Determine the spectrum of a star using unconstrained ls and
constrained inversion'
      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
      call getarg(4,fname_out3)     ! read output file1 name
   endif

   !!read input data
   m=15;n=100
   allocate(s(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

 !!unconstrained LS,ill conditioned
 allocate(invA(n,n))
 call matrixinv(matmul(transpose(A),A),invA,n,n)
 s=matmul(invA,matmul(transpose(A),y))
 !write the results to outputfile1
 print *, 'Writing text to ', fname_out1
 open(22,file=fname_out1) ! open output file
 do i = 1,n
    write(22,*) l(i),' ',s(i)
 enddo
 close(22)
 print *,'unconstrained least squares: jobs done!'

 !!Constrained linear inversion
 ! Construct H
 allocate(H(n,n))
 do i=1,n
    do j=1,n
       if(i==j)then
          H(i,j)=1.0;
       else
          H(i,j)=0.0
       endif
    enddo
 enddo
 !multiplier gamma
 k=250
 allocate(gamma(k))
 allocate(err(k))
 allocate(y_pred(m))
 do i=1,k
    gamma(i)=0.01+(i-1)*0.1
 enddo
 do i=1,k
    call matrixinv(matmul(transpose(A),A)+gamma(i)*H,invA,n,n)
    s=matmul(invA,matmul(transpose(A),y))
```

```
        y_pred=matmul(A,s)
        err(i)=sqrt(sum((y_pred-y)**2)/15)
     enddo
     !write the RMS vs. gamma results to outputfile1
     print *, 'Writing text to ', fname_out2
     open(22,file=fname_out2) ! open output file
     do i = 1,k
        write(22,*) gamma(i),' ',err(i)
     enddo
     close(22)
     print *,'RMS calculation: jobs done!'

     call matrixinv(matmul(transpose(A),A)+25*H,invA,n,n)
     s=matmul(invA,matmul(transpose(A),y))
     print *, 'Writing text to ', fname_out3
     open(22,file=fname_out3) ! open output file
     do i = 1,n
        write(22,*) l(i),' ',s(i)
     enddo
     close(22)
     print *,'Constrained inversion calculation: jobs done!'

     deallocate(y,y_pred,s,A,invA,H,c,l,gamma,err)

endprogram inversion
```