SOME GENERAL STUDY ON IMAGE RESTORATION

by

Asit Kanrar

Under the supervision of

Dr. K. S. Roy

A C K N O W L E D G E M E N T

I express my sincere gratitude to my supervisor Dr. K. S. Roy of Electronics and Computer Science Unit , of Indian Statistical Institute , Calcutta for his excellent guidance , suggestions and constant inspiration given to me for the preparation of this dissertation.

I also acknowledge with thanks the co-operation extended to me by other staffs of Indian Statistical Institute , Calcutta.

*****

C O N T E N T S

1. Introduction

Image restoration is a well known problem of image processing. Several works have been developed in this direction by several researchers [1] [2] [3] [4]. But one drawback of all these works is that they have not considered the nonnegativity constraint of the restored gray level value. The negative values in the gray level value implies an absurdity of negative intensities of radiant energy in the original object distribution.

Another drawback of some of the said works, where certain optimal conditions are tested, is that there is no explicit test for sufficiency of optimality of the restoration. In addition to this sometimes the necessary extreme point of restoration, is shifted heuristically to achieve nonnegative value of the gray levels of the restored image.

Considering all these drawbacks of the existing image restoration methods , in this dissertation , we represent an approach for unconstrained and constrained image restoration methods using **Quadratic Programming Technique**. Along with the suitable constraint equations , needed for improving the quality of the restored image, we consider the nonnegativity constraints of the restored gray level image. We also consider

the necessary and sufficient conditions of optimality of the objective function.

The image 'f', represented by a vector of order (n x 1) is degraded due to defocusing and additive noise. The obtained image 'g', represented by a vector of order (m x 1) is formed by the relation,

$$g = Hf + \eta \quad ,$$

where H is the degraded matrix of order (m x n). H is formed by the concept of point spread function [2] [4] and η of order (m x 1) is uncorrelated random noise. In the absence of any knowledge about η our object is

$$\text{to minimize } Z(f) = (g - Hf)'(g - hf) \quad .....(1)$$

subject to the condition $f \geq \emptyset$.

To further improve the quality of image we add the following constraint equations,

$$Vf \leq P$$

where P is the vector formed by the second derivative of the given image 'g' and 'V' is the smoothing matrix [1].

We also propose a method to estimate the degradation matrix H on the basis of sample values of original image 'f' and degraded image 'g'.

2. Statement of the problem

Minimize $Z(f) = (g - Hf)'(g - Hf)$

subject to $f \geq \underline{\emptyset}$

and $Vf \leq P$

wher $f=(f_1, f_2, \ldots f_n)$ and $P=(b_1, b_2, \ldots b_m)$

i.e. We can rewrite the objective function $Z(f)$ as

$$Z(f) = g'g - 2g'Hf + f'H'Hf$$

$$= \hat{g} - Cf + f'Df$$

where $\hat{g}$ is scalar and $C = 2g'H$, $D = H'H$

Therefore we can rewrite the problem as follows,

Minimize $Z(f) = \hat{g} - Cf + f'Df$

subject to the condition, $f \geq \underline{\emptyset}$

and $Vf \leq P$

The function $f'Df$ defines a quadratic form where D is symmetric. If the matrix D is positive definite , $Z(f)$ is strictly convex in 'f' for minimization. The constraints are linear in this case. It guarantees a convex solution space.

The solution to this problem is secured by applying Kuhn-Tucker necessary condition (Page - 559 , [5]). Since the

solution space is a convex set (as the constraint equations are linear), the sufficient conditions for a global minimum is obtained if Z(f) is strictly convex. Now the entire problem may be written as,

$$\text{Minimize, } Z(f) = g - Cf + f'Df$$

$$\text{subject to, } G(f) = \begin{pmatrix} V \\ -I \end{pmatrix} f - \begin{pmatrix} P \\ \emptyset \end{pmatrix} \leq \emptyset$$

Let $\lambda = (\lambda_1, \lambda_2, \ldots \lambda_m)'$ and $U = (\mu_1, \mu_2, \ldots \mu_n)'$ be the Lagrangean multipliers corresponding to the two set of constraints $Vf - P \leq \emptyset$ and $-f \leq \emptyset$, respectively.

Application of the Kuhn-Tucker conditions yield

$$\lambda \geq \emptyset$$

$$U \geq \emptyset$$

$$\nabla Z(f) + (\lambda' \quad U') \nabla G(f) = \emptyset$$

$$\lambda_i \left(b_i - \sum_{j=1}^{n} C_{ij} f_j\right) = \emptyset, \quad i = 1, 2, \ldots m$$

$$\mu_j f_j = \emptyset, \quad j = 1, 2, \ldots n$$

$$Vf \leq P$$

$$-f \leq \emptyset$$

Now $Z(f) = -C + 2f'D$

and $G(f) = \begin{pmatrix} V \\ -I \end{pmatrix}$

Let $S = P - Vf$ be the slack variables then we obtain,

$$2\,f'D + V - U' = C$$

$$\lambda_i S_i = \emptyset, \quad i = 1,2, \ldots m$$

$$\mu_j f_j = \emptyset, \quad j = 1,2, \ldots n$$

$$Vf + S = P$$

$$\lambda, U, f, S \geq \emptyset$$

Taking the transpose , the first set of equation may be rewritten as,

$$2\,Df' + V' - U = C' \quad (D \text{ is symmetric})$$

Hence the necessary conditions may be expressed as ,

$$\begin{pmatrix} 2D & V' & -I & \emptyset \\ V & \emptyset & \emptyset & I \end{pmatrix} \begin{pmatrix} f \\ \lambda \\ U \\ S \end{pmatrix} = \begin{matrix} C' \\ P \end{matrix}$$

$$\lambda_i S_i = \emptyset \quad i = 1,2..m$$

$$\mu_j f_j = \emptyset \quad j = 1,2..n$$

$$\lambda, U, f, S \geq \emptyset$$

In case if we do not consider the quality constraint

$$Vf \leq P$$

the necessary condition will be reduced to

$$\begin{pmatrix} 2D & -I \end{pmatrix} \begin{pmatrix} f \\ U \end{pmatrix} = C'$$

$$\lambda_j f_j = \emptyset, \quad j = 1,2 \ldots n$$

$$f \geq \emptyset$$

Except for the conditions $\mu_i f_i = 0 = \lambda_i S_i$ the remaining equations are linear in f , λ , U and S. Therfore we require to solve a set of linear equation while satisfying the additional conditions $\mu_i f_i = 0 = \lambda_i S_i$. Since the solution space is convex, the feasible solution satisfying all these conditions must give the optimum solution, provided Z(f) is strictly convex.

The solution of the above system is obtained by using Phase I of the two phase method of simplex algorithm for linear programming. The only restriction is that the condition $\mu_i f_i = 0 = \lambda_i S_i$ should be maintained always. This means if λ_i is in the basic solution at a positive level, S_i cannot become basic at positive level. Similarly μ_i and f_i cannot become positive simultaneously. The systematic procedure of simplex, for checking the feasibility of the solution space , is to introduce appropriate number of artificial variables to the set of constraint equations. The Phase I of simplex iteration will end in the usual manner with the sum of artificial variables equal to zero.

3. Identification of matrix H

Let us consider the following linear degradation process represented by Fig. 1.

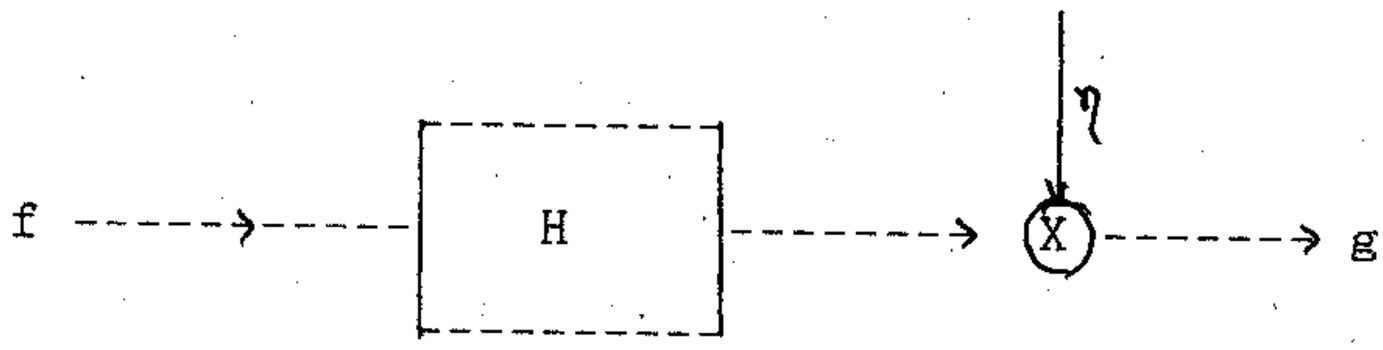


Fig. 1.

Given 'f' (the training image sample) and 'g' (the measured image signal) we are to identify the degradation process H. Depending upon the quality of the random noise at different time interval we will get different 'g' for the same given 'f'. For computational simplicity initially we ignore the random noise term from the generalised representation of the degradation process.

i.e. we consider,

$$g = Hf$$

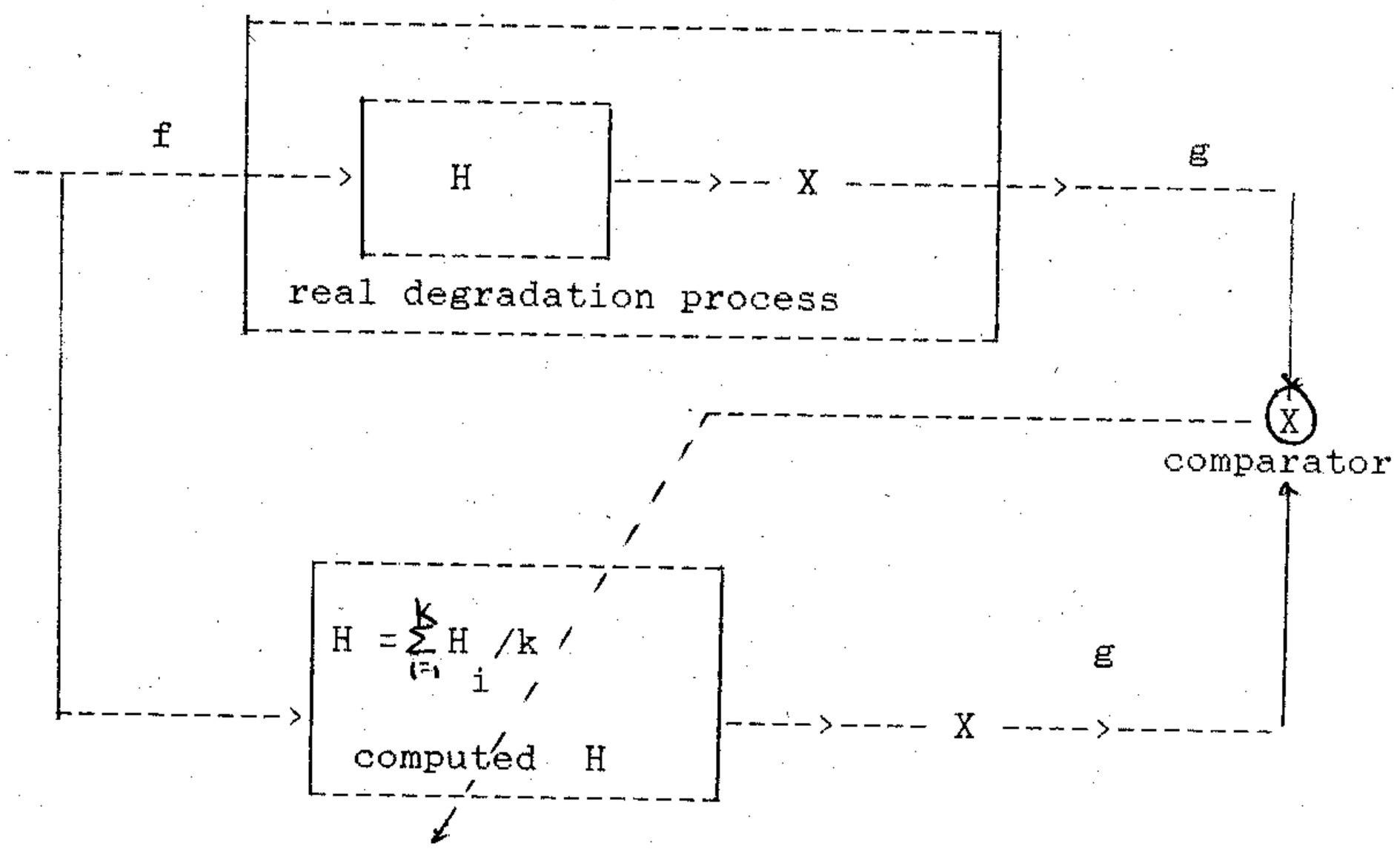
Now at any particular time instant k if we have g_k, then, the degradation process H_k at the time instant k can be represented as,

$$H_k = f^{+} g_k$$

where f^{+} is the Penrose Inverse of the given training sample f. Now the crude estimate of H which will be ultimately modulated at the output channel by the random noise is,

$$H = \sum_{i=1}^{k} H_i / k$$

In this connection of H identification some adaptive scheme is proposed as follows,



Further upgradation of H

Fig. 2.

4. Results and Discussions

To implement the algorithms (i) image restoration considering quality constraint and (ii) image restoration without considering quality constraint the degradation matrix H ( ) is taken as follows ,

$$H = \begin{bmatrix} .5 & .25 & \emptyset & \emptyset & \cdots\cdots\cdots & \emptyset \\ .25 & .5 & .25 & \emptyset & \cdots\cdots\cdots & \emptyset \\ \emptyset & .25 & .5 & .25 & \cdots\cdots\cdots & \emptyset \\ & & & & & \\ & & & & & \\ \cdots\cdots\cdots & & & .25 & .5 & .25 & \emptyset \\ \cdots\cdots\cdots & & & \emptyset & .25 & .5 & .25 \\ \cdots\cdots\cdots & & & \emptyset & \emptyset & .25 & .5 \end{bmatrix}$$

The ANSI-77 FORTRAN Program listing along with their outputs in terms of gray level values are given in Appendix-A and Appendix-B respectively.

In both cases to show the effectiveness of the algorithm the gray level values of (i) an original image (ii) degrading it due to the effect of degradation plus a random noise and then (iii) restoring the image from the degraded image using

the algorithms have been shown.

In Appendix-C a program listing and its output have been included , where instead of assuming matrix H it has been estimated on some sample values of ´f´ and ´g´ and the image restoration has been implemented considering that estimated value of H.

In all the three cases the results we obtain are quite satisfactory but one disadvantage of this algorithm is that it requires a very large computation due to the large image matrix dimension which further requires a huge computer memory.

```
c *** IMAGE RESTORATION CONSIDERING QUALITY CONSTRAINT ***
c *** Input : AR - Original Image Matrix in character form
c ***           IP _ Image Matrix converted to gray level value
c ***           F  - Image converted to vector form
c ***           H  - Degraded matrix
c ***           V  - Smoothness matrix
c
c *** Output  G  - Degraded Image in vector form
c ***           F  - Restored Image in vector form
c
        character*1 ar(64,64)
c
        dimension ip(64,64),f(4Ø96),h(256,256),p(4Ø96)
        dimension c(256),d(256,256),v(256,256)),g(4Ø96)
c
c *** SET UP ***
c
        open(unit=5,file='temp1.dat')
        open(unit=6,file='prn:')
        read(5,*)m1,n1
c
      do 5 i=1,m1
    5 read(5,*) (ar(i,j),j=1,n1)
c
          call convrt(m1,n1,ar,ip)
c
          do 6 i=1,m1
    6 write(6,1) (ip(i,j),j=1,n1)
    1 format(1x,16i3)
c
          call image(m1,n1,ip)
c
c *** CONVERT THE MATRIX IN VECTOR FORM ***
c
        l=1
        do 1Ø i=1,m1
        do 1Ø j=1,n1
        f(l)=ip(i,j)
   1Ø l=l+1
c
          n=m1*n1
c
c *** FORMATION OF DEGRADATION MATRIX ***
c
          do 2Ø i=1,n
          do 2Ø j=1,n
   2Ø h(i,j)=Ø.
c
          h(1,1)=.5
          h(1,2)=.25
```

```
c
      do 25 i=2,n
      h(i,i)=.5
      h(i,i-1)=.25
   25 h(i,i+1)=.25
c
      read(5,*) m2,n2
c
      m=m2*n2
c
      do 3Ø i=1,m
      g(i)=Ø.
      do 3Ø j=1,n
   3Ø g(i)=g(i)+h(i,j)*f(j)
c
      l=1
      do 35 i=1,m2
      do 35 j=1,n2
      ip(i,j)=g(l)+.5
   35 l=l+1
c
      do 4Ø i=1,m2
   4Ø write(6,1) (ip(i,j),j=1,n2)
c
      call image(m2,n2,ip)
c
      l=1
      do 57 i=1,m2
      do 57 j=1,n2
      ig(l)=ip(i,j)
   57 l=l+1
c
c *** COMPUTATION OF MATRIX D AND VECTOR C ***
c
      do 6Ø i=1,n
      c(i)=Ø.
      do 6Ø j=1,m
   6Ø c(i)=c(i)+2.*ig(j)*h(j,i)
c
      do 65 i=1,n
      do 65 j=i,n
      d(i,j)=Ø.
      do 65 k=1,m
   d(i,j)=d(i,j)+h(k,i)*h(k,j)
      if (i .eq. j) go to 65
      d(j,i)=d(i,j)
   65 continue
c
      do 75 i=1,n
      do 75 j=1,n
```

```
   75 v(i,j)=0.
c
          v(1,1)=1.
          v(2,1)=-2.
          v(2,2)=1.
c
          do 76 i=3,n
          v(i,i-2)=1.
          v(i,i-1)=-2.
          v(i,i)=1.
   76 continue
c
          do 78 i=1,m
          p(i)=0.
          do 78 k=1,m
   78 p(i)=p(i)+v(i,k)*ig(k)
c
          CALL qfm(m,n,d,v,c,p,f)
c
c *** FORM THE IMAGE MATRIX IP ***
c *** FROM THE VECTOR F ***
c
          l=1
          do 80 i=1,m1
          do 80 j=1,n1
          if (f(l) .gt. 31.) f(l)=31.
          ip(i,j)=f(l)+.5
   80 l=l+1
c
          do 90 i=1,m1
   90 write(6,1) (ip(i,j),j=1,n1)
c
c *** CALL THE IMAGE SUBROUTINE
c *** TO PRINT THE IMAGE
c
          CALL image(m1,n1,ip)
c
          stop
          end
```

```
c *** SUBROUTINE TO FORM THE QUADRATIC MODEL ***
c
      SUBROUTINE qfm(m,n,d,v,c,p,f)
c
c *** FORMATION FOR SIMPLEX ***
c
      dimension v(256,256),d(256,256),c(256),a(515,154Ø)
      dimension ib(256),isv(154Ø),p(256),f(256)
c
      do 1Ø i=1,n
      do 1Ø j=i,n
    a(i,j)=2.*d(i,j)
      if (i .eq. j) go to 1Ø
      a(j,i)=a(i,j)
 1Ø continue
c
      n1=n+m
      n2=2*n+m
c
      do 2Ø i=1,n
      do 2Ø j=1,m
      k=n+j
    a(i,k)=v(j,i)
 2Ø a(k,i)=v(j,i)
c
      do 25 i=1,m
      do 25 j=1,m
      i1=n+i
      j1=n+j
 25 a(i1,j1)=Ø.
c
      do 3Ø i=1,n1
      do 3Ø j=1,n
      j1=n1+j
      j2=n2+j
      if (i .eq. j) go to 26
      a(i,j1)=Ø.
      a(i,j2)=Ø.
      go to 3Ø
 26 a(i,j1)=-1.
      a(i,j2)=1.
 3Ø continue
c
      n2=3*n+m
c
      do 4Ø i=1,n
      do 4Ø j=1,m
      j1=n2+j
 4Ø a(i,j1)=Ø.
c
```

```
      do 45 i=1,m
      do 45 j=1,m
      i1=n+i
      j1=n2+j
      if (i .eq. j) then
            a(i1,j1)=1.
         else
            a(i1,j1)=0.
      endif
   45 continue
c
c *** TO FORM THE RHS VECTOR ***
c
      n3=3*n+2*m+1
c
      do 50 i=1,n
   50 a(i,n3)=c(i)
c
      do 55 i=1,m
      i1=n+i
   55 a(i1,n3)=p(i)
c
c *** TO FORM THE COST VECTOR ***
c
      n2=n+m+1
c
      do 60 j=1,n3
   60 a(n2,j)=0.
c

      ni=2*n+m+1
      nj=3*n+m
      nk=3*n+2*m
c
      do 70 j=ni,nj
   70 a(n2,j)=-1.
c
      do 80 j=1,n3
      do 80 i=1,n
   80 a(n2,j)=a(n2,j)+a(i,j)
c
c *** TO FORM THE BASIS ***
c
      do 100 i=1,nk
  100 isv(i)=0
c
      do 110 i=1,n1
      ib(i)=ni
      isv(ni)=1
  110 ni=ni+1
```

```
c
c *** TO CHECK IF ANY ELEMENT OF ***
c *** RHS VECTOR IS NEGATIVE     ***
c *** IF SO , MODIFY THE SIMPLEX ***
c *** MATRIX AND BASIS           ***
c
         k=Ø
         do 14Ø i=1,n1
         if(a(i,n3) .ge. Ø.) go to 14Ø
         n31=n3+1
         leave=i
         k=k+1
c
         do 15Ø i1=1,n2
         a(i1,n31)=a(i1,n3)
 15Ø a(i1,n3)=Ø.
c
         do 16Ø j=1,n31
 16Ø a(leave,j)=-a(leave,j)
c
         a(leave,n3)=1.
         a(n2,n3)=-1.
c
         do 17Ø j=1,n31
 17Ø a(n2,j)=a(n2,j)+a(leave,j)
c
         ileave=ib(leave)
         ib(leave)=n3
         isv(ileave)=Ø
         isv(n3)=1
         n3=n3+1
c
 14Ø continue
c
         nl=nk+k
c
         CALL qsm(k,m,n,ib,isv,a)
c
c *** TO FORM THE RESTORED VECTOR ***
c *** FROM THE BASIS              ***
c
         do 12Ø i=1,n
 12Ø f(i)=Ø.
         do 13Ø i=1,n1
         if (ib(i) .gt. n) go to 13Ø
         l=ib(i)
         f(l)=a(i,n3)
 13Ø continue
         return
         end
```

```
c *** SUBROUTINE FOR SIMPLEX PHASE 1.  ***
c *** WITH ADDITIONAL RESTRICTION FOR  ***
c *** QUADRATIC PROGRAMMING            ***
c
      SUBROUTINE qsm(k,m,n,ib,isv,a)
c
      dimension ib(256),isv(154Ø),a(515,154Ø),b(154Ø)
c
      n1=n+m
      n2=n+m+1
      n3=3*n+2*m+k+1
      ni=2*n+m+1
      nj=3*n+m
      nk=3*n+2*m
      nl=nk+k
c
      iter=Ø
    5 iter=iter+1
c
      do 1Ø j=1,nl
   1Ø b(j)=a(n2,j)
c
   2Ø iev=Ø
         cev=.ØØØØ1
c
         do 3Ø j=1,nl
         if(b(j) .le. cev) go to 3Ø
         iev=j
         cev=b(j)
   3Ø continue
c
         if (cev .le. .ØØØØ1) go to 15Ø
c
c *** CHECK WHETHER CORRESPONDING ***
c *** LAGRANGE'S MULTIPLIER IS    ***
c *** IN THE BASIS                ***
c
         if (iev .lt. n+1) go to 6Ø
         if (iev .lt. n2) go to 4Ø
         if (iev .lt. ni) go to 7Ø
         if ((iev .gt. nj) .and. (iev .le. nk)) go to 5Ø
         go to 1ØØ
c
   4Ø i=iev+2*n+m
      if (isv(i) .eq. 1) go to 8Ø
         go to 1ØØ
c
   5Ø i=iev-2*n-m
      if (isv(i) .eq. 1) go to 8Ø
```

```
            go to 100
c
   60 i=iev+n1
            if (isv(i) .eq. 1) go to 80
            go to 100
c
   70 i=iev-n1
            if (isv(i) .eq. 1) go to 80
            go to 100
c
   80 b(iev)=0.
            go to 20
c
c *** CHECK FOR UNBOUNDED SOLUTION ***
c
  100 spr=10.e10
c
            do 110 i=1,n1
            if (a(i,iev) .lt. .00001) go to 110
            pr=a(i,n3)/a(i,iev)
            if (pr .ge. spr) go to 110
            spr=pr
            leave=i
  110 continue
c
            if (spr .le. 10.e10) go to 120
            write(*,1)
    1 format(' unbounded solution')
            stop
c
  120 isv(iev)=1
            i=ib(leave)
            isv(i)=0
            ib(leave)=iev
c
c *** NEW BASIS AND SOLUTION VECTOR ***
            pivot=a(leave,iev)
c
            do 125 j=1,n3
  125 a(leave,j)=a(leave,j)/pivot
c
            do 130 i=1,n2
            if (i .eq. leave) go to 130
            hold=a(i,iev)
            do 140 j=1,n3
  140 a(i,j)=a(i,j)-hold*a(leave,j)
  130 continue
    3 format(1x,8(1x,f8.5))
c
            go to 5
```

```
c
  150 if (a(n2,n3) .gt. .00001) go to 160
c
          return
c
  160 write(6,2)
    2 format(' no feasible solution')
c
          stop
          end
```

```
      SUBROUTINE convrt(m,n,ar,ia)
c
c *** SUBROUTINE TO CONVERT THE GRAY LEVEL ***
c *** CHARACTER DATA TO INTEGER DATA       ***
c
      character*1 ar(64,64)
      dimension ia(64,64)
c
      do 10 i=1,m
      do 10 j=1,n
      if ((ar(i,j) .ge. '0') .and. (ar(i,j) .le. '9')) then
         ia(i,j)=ichar(ar(i,j))-ichar('0')
      else
         ia(i,j)=ichar(ar(i,j))-ichar('A')+10
      endif
   10 continue
c
      return
      end
```

```
      SUBROUTINE image(nx,ny,ib)
c
c *** SUBROUTINE TO PRINT THE IMAGE ***
            dimension ib(64,64),iblank(5)
            character*1 line(128,5),gray(32,5)
c
c *** SPECIFY GRAY-LEVEL CHARACTERS ***
c
            data gray /6*'M',5*'H','X','H','X','O',
     1 'Z','W','M','N','O','S','=','I','*','+',
     2 '+','=',':','-','.',' ','-',' ',' ',' ',
     3 '*','+','+',4*'-',5*' ','=',' ',' ',' ',' ',
     4 3*' ','-',3*' ',4*'#','Ø','O','+','-',
     5 24*' ','O','Ø','O',29*' ','+',31*' '/
c
            ix=nx
            iy=2*ny
            do 21Ø i=1,ix
            do 19Ø j=1,5
  19Ø iblank(j)=Ø
            do 2ØØ k=2,iy,2
            j=k/2
            ng=32-ib(i,j)
            do 2ØØ l=1,5
            line(k-1,l)=gray(ng,l)
            line(k,l)=gray(ng,l)
            if (ng .ne. 32) iblank(l)=1
  2ØØ continue
            write(6,2)
            do 21Ø l=1,5
            if (iblank(l) .eq. Ø) go to 21Ø
            write(6,3) (line(m,l),m=1,iy)
  21Ø continue
    2 format(' ')
    3 format('+',3x,128a1)
            return
            end
```

Ø	Ø	Ø	4
Ø	Ø	4	21
2Ø	24	2Ø	22
7	15	24	2Ø

Gray Level Value of The Original Image

Ø	Ø	1	2
1	1	7	17
21	22	22	18
13	15	21	16

Gray Level Value of The Degraded Image

Ø	Ø	1	2
1	1	7	17
21	22	22	18
13	15	21	16

Gray Level Value of The Restored Image

```
c ***  IMAGE RESTORATION WITHOUT CONSIDERING QUALITY  CONSTRAINT
c ***  Input : AR - Original Image Matrix in character form
c ***            IP - Image Matrix converted to gray level value
c ***            F  - Image converted to vector form
c ***            H  - Degraded matrix
c
c ***  Output  G  - Degraded Image in vector form
c ***            F  - Restored Image in vector form
c
        character*1 ar(64,64)
c
        dimension ip(64,64),f(4Ø96),h(256,256)
        dimension c(256),d(256,256)
        dimension ig(4Ø96),g(4Ø96),p(4Ø96)
c
c *** SET UP ***
c
        open(unit=5,file='temp1.dat')
        open(unit=6,file='prn:')
        read(5,*)m1,n1
c
      do 5 i=1,m1
    5 read(5,*) (ar(i,j),j=1,n1)
c
           call convrt(m1,n1,ar,ip)
c
           do 6 i=1,m1
    6 write(6,1) (ip(i,j),j=1,n1)
    1 format(1x,16i3)
c
           call image(m1,n1,ip)
c
c *** CONVERT THE MATRIX IN VECTOR FORM ***
c
           l=1
           do 1Ø i=1,m1
           do 1Ø j=1,n1
           f(l)=ip(i,j)
   1Ø l=l+1
c
           n=m1*n1
c
c *** FORMATION OF DEGRADATION MATRIX ***
c
           do 2Ø i=1,n
           do 2Ø j=1,n
   2Ø h(i,j)=Ø.
c
           h(1,1)=.5
           h(1,2)=.25
```

```
c
      do 25 i=2,n
      h(i,i)=.5
      h(i,i-1)=.25
   25 h(i,i+1)=.25
c
      read(5,*) m2,n2
c
      m=m2*n2
c
      do 3Ø i=1,m
      g(i)=Ø.
      do 3Ø j=1,n
   3Ø g(i)=g(i)+h(i,j)*f(j)
c
      l=1
      do 35 i=1,m2
      do 35 j=1,n2
      ip(i,j)=g(l)+.5
   35 l=l+1
c
      do 4Ø i=1,m2
   4Ø write(6,1) (ip(i,j),j=1,n2)
c
      call image(m2,n2,ip)
c
      l=1
      do 57 i=1,m2
      do 57 j=1,n2.
      ig(l)=ip(i,j)
   57 l=l+1
c
c *** COMPUTATION OF MATRIX D AND VECTOR C ***
c
      do 6Ø i=1,n
      c(i)=Ø.
      do 6Ø j=1,m
   6Ø c(i)=c(i)+2.*ig(j)*h(j,i)
c
      do 65 i=1,n
      do 65 j=i,n
      d(i,j)=Ø.
      do 65 k=1,m
   d(i,j)=d(i,j)+h(k,i)*h(k,j)
      if (i .eq. j) go to 65
      d(j,i)=d(i,j)
   65 continue
c
      CALL qfm1(n,d,c,p,f)
c
```

```
c *** FORM THE IMAGE MATRIX IP ***
c *** FROM THE VECTOR F ***
c
          l=1
          do 8Ø i=1,m1
          do 8Ø j=1,n1
          if (f(l) .gt. 31.) f(l)=31.
          ip(i,j)=f(l)+.5
    8Ø l=l+1
c
          do 9Ø i=1,m1
    9Ø write(6,1) (ip(i,j),j=1,n1)
c
c *** CALL THE IMAGE SUBROUTINE
c *** TO PRINT THE IMAGE
c
          CALL image(m1,n1,ip)
c
          stop
          end
```

```
c *** SUBROUTINE TO FORM THE QUADRATIC MODEL ***
c
      SUBROUTINE qfm1(n,d,c,p,f)
c
c *** FORMATION FOR SIMPLEX ***
c
      dimension v(256,256),d(256,256),c(256),a(515,154Ø)
      dimension ib(256),isv(154Ø),p(256),f(256)
c
      do 1Ø i=1,n
      do 1Ø j=i,n
    a(i,j)=2.*d(i,j)
      if (i .eq. j) go to 1Ø
      a(j,i)=a(i,j)
   1Ø continue
c
      n1=n
      n2=2*n.
c
      do 3Ø i=1,n1
      do 3Ø j=1,n
      j1=n1+j
      j2=n2+j
      if (i .eq. j) go to 26
      a(i,j1)=Ø.
      a(i,j2)=Ø.
      go to 3Ø
   26 a(i,j1)=-1.
      a(i,j2)=1.
   3Ø continue
c
c *** TO FORM THE RHS VECTOR ***
c
      n3=3*n+1
c
      do 5Ø i=1,n
   5Ø a(i,n3)=c(i)
c
c *** TO FORM THE COST VECTOR ***
c
      n2=n+1
c
      do 6Ø j=1,n3
   6Ø a(n2,j)=Ø.
c
      ni=2*n+1
      nj=3*n
      nk=3*n
c
      do 7Ø j=ni,nj
```

```
   70 a(n2,j)=-1.
c
         do 80 j=1,n3
         do 80 i=1,n
   80 a(n2,j)=a(n2,j)+a(i,j)
c
         do 100 i=1,nk
  100 isv(i)=0
c
         do 110 i=1,n1
         ib(i)=ni
         isv(ni)=1
  110 ni=ni+1
c
         CALL qsm1(n,ib,isv,a)
c *** TO FORM THE RESTORED VECTOR ***
c *** FROM THE BASIS               ***
c
         do 120 i=1,n
  120 f(i)=0.
c
         do 130 i=1,n1
         if (ib(i) .gt. n) go to 130
         l=ib(i)
         f(l)=a(i,n3)
  130 continue
c
         return
         end
```

```
c *** SUBROUTINE FOR SIMPLEX PHASE 1. ***
c *** WITH ADDITIONAL RESTRICTION FOR ***
c *** QUADRATIC PROGRAMMING           ***
c
      SUBROUTINE qsm1(m,n,ib,isv,a)
c
      dimension ib(256),isv(154Ø),a(515,154Ø),b(154Ø)
c
      n1=n
      n2=n+1
      n3=3*n+1
      ni=2*n+1
      nj=3*n
      nk=3*n
      nl=nk
c
      iter=Ø
    5 iter=iter+1
c
      do 1Ø j=1,nk
   1Ø b(j)=a(n2,j)
c
   2Ø iev=Ø
      cev=.ØØØØ1
c
      do 3Ø j=1,nk
      if(b(j) .le. cev) go to 3Ø
      iev=j
      cev=b(j)
   3Ø continue
c
      if (cev .le. .ØØØØ1) go to 15Ø
c
c *** CHECK WHETHER CORRESPONDING ***
c *** LAGRANGE'S MULTIPLIER IS    ***
c *** IN THE BASIS                ***
c
      if (iev .lt. n+1) go to 6Ø
      if (iev .lt. ni) go to 7Ø
      go to 1ØØ
c
   6Ø i=iev+n1
      if (isv(i) .eq. 1) go to 8Ø
      go to 1ØØ
c
   7Ø i=iev-n1
      if (isv(i) .eq. 1) go to 8Ø
      go to 1ØØ
c
   8Ø b(iev)=Ø.
```

```
            go to 20
c
c *** CHECK FOR UNBOUNDED SOLUTION ***
c
  100 spr=10.e10
c
            do 110 i=1,n1
            if (a(i,iev) .lt. .00001) go to 110
            pr=a(i,n3)/a(i,iev)
            if (pr .ge. spr) go to 110
            spr=pr
            leave=i
  110 continue
c
            if (spr .le. 10.e10) go to 120
            write(*,1)
    1 format(' unbounded solution')
            stop
c
  120 isv(iev)=1
            i=ib(leave)
            isv(i)=0
            ib(leave)=iev
c
c *** NEW BASIS AND SOLUTION VECTOR ***
c
            pivot=a(leave,iev)
c
      .     do 125 j=1,n3
  125 a(leave,j)=a(leave,j)/pivot
c
            do 130 i=1,n2
            if (i .eq. leave) go to 130
            hold=a(i,iev)
            do 140 j=1,n3
  140 a(i,j)=a(i,j)-hold*a(leave,j)
  130 continue
    3 format(1x,8(1x,f8.5))
c
            go to 5
c
  150  if (a(n2,n3) .gt. .00001) go to 160
c
       return
c
  160 write(6,2)
    2 format(' no feasible solution')
c
            stop
            end
```

Ø	Ø	Ø	4
Ø	Ø	4	21
2Ø	24	2Ø	22
7	15	24	2Ø

Gray Level Value of The Original Image

Ø	Ø	1	2
1	1	7	17
21	22	22	18
13	15	21	16

Gray Level Value of The Degraded Image

Ø	Ø	Ø	4
Ø	Ø	3	22
2Ø	21	25	17
13	9	29	18

Gray Level Value of The Restored Image

```
c *** IMAGE RESTORATION CONSIDERING QUALITY CONSTRAINT ***
c *** WITH AN ESTIMATED DEGRADATION MATRIX                ***
c *** Input : ar - Original Image Matrix in character form
c ***         ip - Image Matrix converted to gray level value
c ***         f  - Image converted to vector form
c ***         v  - Smoothness Matrix
c
c *** Output  g  - Degraded Image in vector form
c ***         f  - Restored Image in vector form
c
      character*1 ar(64,64)
c
      dimension ip(64,64),finv(4Ø96),ig(4Ø96),g(4Ø96)
      dimension f(16),c(16),d(16,16),v(16,16),p(16),h(16,16)
      dimension ib(35),isv(1ØØ),a(35,1ØØ),b(1ØØ)
c
c *** SET UP ***
c
      open(unit=5,file='temp.dat')
      open(unit=6,file='prn:')
      read(5,*)m1,n1
c
      do 5 i=1,m1
    5 read(5,*) (ar(i,j),j=1,n1)
c
      call convrt(m1,n1,ar,ip)
c
      do 6 i=1,m1
    6 write(6,1) (ip(i,j),j=1,n1)
    1 format(1x,16i3)
c
      call image(m1,n1,ip)
c
c *** CONVERT THE MATRIX IN VECTOR FORM ***
c
      l=1
      do 1Ø i=1,m1
      do 1Ø j=1,n1
      f(l)=ip(i,j)
   1Ø l=l+1
c
      n=m1*n1
c
c *** COMPUTATION OF THE PENROSE INVERSE ***
c *** OF THE VECTOR F. i.e. FINV=F/(F'F) ***
c
      isum=Ø
      do 2Ø i=1,n
   2Ø isum=isum+f(i)*f(i)
c
```

```
c *** ISUM IS ACTUALLY F'F ***
c
          do 3Ø i=1,n
    3Ø finv(i)=f(i)/isum
c
          write(6,2) (finv(i),i=1,n)
     2 format(' penrose inverse follows'/1x,16(1x,f7.4))
c
          read(5,*) m2,n2
c
          do 35 i=1,m2
    35 read(5,*) (ar(i,j),j=1,n2)
c
          call convrt(m2,n2,ar,ip)
c
          call image(m2,n2,ip)
c
          l=1
          do 45 i=1,m2
          do 45 j=1,n2
          g(l)=ip(i,j)
    45 l=l+1
c
          m=m2*n2
c
c *** COMPUTATION OF DEGRADATION ***
c
          do 5Ø i=1,m
          do 5Ø j=1,n
    5Ø h(i,j)=finv(j)*g(i)
c
          write(6,16)
    16 format(' degradation process follows')
          do 55 i=1,m
    55 write(6,17) (h(i,j),j=1,n)
    17 format(1x,16(1x,f7.4))
c
          do 56 i=1,m2
    56 read(5,*) (ar(i,j),j=1,n2)
c
          call convrt(m2,n2,ar,ip)
c
          call image(m2,n2,ip)
c
          l=1
          do 57 i=1,m2
          do 57 j=1,n2
          ig(l)=ip(i,j)
    57 l=l+1
c
```

```
c *** COMPUTATION OF MATRIX D AND VECTOR C ***
c
          do 6Ø i=1,n
          c(i)=Ø.
          do 6Ø j=1,m
    6Ø c(i)=c(i)+2.*ig(j)*h(j,i)
c
          do 65 i=1,n
          do 65 j=i,n
          d(i,j)=Ø.
          do 65 k=1,m
       d(i,j)=d(i,j)+h(k,i)*h(k,j)
          if (i .eq. j) go to 65
          d(j,i)=d(i,j)
    65 continue
c
          write(6,7) (c(i),i=1,n)
     7 format(' vector c'/1x,16(1x,f7.4))
c
          write(6,8)
     8 format(' matrix d')
c
          do 7Ø i=1,n
    7Ø write (6,9) (d(i,j),j=1,n)
     9 format(1x,16(1x,f7.4))
c
          CALL qfm(m,n,d,v,c,p,f)
c
c *** FORM THE IMAGE MATRIX IP ***
c *** FROM THE VECTOR F ***
c
          l=1
          do 8Ø i=1,m1
          do 8Ø j=1,n1
          if (f(l) .gt. 31.) f(l)=31.
          ip(i,j)=f(l)+.5
    8Ø l=l+1
c
          do 9Ø i=1,m1
    9Ø write(6,*) (ip(i,j),j=1,n1)
c
c *** CALL THE IMAGE SUBROUTINE
c *** TO PRINT THE IMAGE
c
          CALL image(m1,n1,ip)
c
          stop
          end
```

Ø	Ø	Ø	4
Ø	Ø	4	21
2Ø	24	2Ø	22
7	15	24	2Ø

Gray Level Value of Original Image

Ø	Ø	1	2
1	1	7	17
21	22	22	18
13	15	21	16

Gray Level Value of Degraded Image

Ø Ø Ø 4 Ø Ø 4 21 2Ø 24 2Ø 22 7 15 24 2Ø

Gray Level Value of Original Value in Vector form : f

.ØØ1 * (Ø Ø Ø 11 Ø Ø 11 59 56 67 56 61 2Ø 42 67 56)

Penrose inverse of the vector f

$$H = .0001 * \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 1 & 6 & 6 & 7 & 6 & 6 & 2 & 4 & 7 & 6 \\
0 & 0 & 0 & 2 & 0 & 0 & 2 & 12 & 11 & 13 & 11 & 12 & 4 & 8 & 13 & 11 \\
0 & 0 & 0 & 1 & 0 & 0 & 1 & 6 & 6 & 7 & 6 & 6 & 2 & 4 & 7 & 6 \\
0 & 0 & 0 & 1 & 0 & 0 & 1 & 6 & 6 & 7 & 6 & 6 & 2 & 4 & 7 & 6 \\
0 & 0 & 0 & 8 & 0 & 0 & 8 & 41 & 39 & 47 & 39 & 43 & 14 & 29 & 47 & 39 \\
0 & 0 & 0 & 19 & 0 & 0 & 19 & 99 & 95 & 113 & 95 & 104 & 33 & 71 & 114 & 95 \\
0 & 0 & 0 & 23 & 0 & 0 & 23 & 123 & 117 & 141 & 117 & 129 & 41 & 88 & 141 & 117 \\
0 & 0 & 0 & 25 & 0 & 0 & 25 & 129 & 123 & 147 & 123 & 135 & 43 & 92 & 147 & 123 \\
0 & 0 & 0 & 25 & 0 & 0 & 25 & 129 & 123 & 147 & 123 & 125 & 43 & 92 & 147 & 123 \\
0 & 0 & 0 & 20 & 0 & 0 & 20 & 105 & 100 & 121 & 100 & 110 & 35 & 75 & 121 & 100 \\
0 & 0 & 0 & 14 & 0 & 0 & 14 & 76 & 73 & 87 & 73 & 80 & 25 & 54 & 87 & 73 \\
0 & 0 & 0 & 17 & 0 & 0 & 17 & 88 & 84 & 100 & 84 & 92 & 29 & 63 & 100 & 84 \\
0 & 0 & 0 & 23 & 0 & 0 & 23 & 123 & 117 & 141 & 117 & 129 & 41 & 88 & 141 & 117 \\
0 & 0 & 0 & 18 & 0 & 0 & 18 & 94 & 89 & 107 & 89 & 98 & 31 & 67 & 107 & 89
\end{bmatrix}$$

Estimated Value of the matrix H

0	0	0	0
0	0	7	17
22	23	23	20
15	17	24	19

Gray Level Value of The Restored Image

References

1. R. C. Gonzalez and P. Wintz., 'Digital Image Processing', Addison-Wesley Publishing Company , 1977.

2. H. C. Andrews and B. R. Hunt , 'Digital Image Restoration', Prentice-Hall , Inc., 1977.

3. B. R. Hunt , 'The application of constrained least squares estimation to image restoration by digital computer', IEEE Trans. Computers , Vol. C-22 No. 9 PP. 8Ø5-812 , 1973.

4. B. Chanda , B. B. Chaudhuri and D. Dutta Majumdar, 'Application of least square estimation technique for image restoration using signal-noise correlation constraint', IEEE Trans. Syst. Man Cybern., Vol. SMC-14 No. 3 PP. 515-519 , 1984.

5. H. Taha , 'Operations Research - An Introduction', Macmillan Publishing Co. , Inc. , 1982.