Appendix, GRNN

```
C       This program is far from being a commercial program
C       please feel free to change and expand the program
C       in any way you want.
C
C       Things you could do:
C         Link in a normaization program to have normalized data
C         Link a subroutine in to shown development of the SSQ
C         Link a graphing program in to show predictions
C
C       Have fun doing that
C

C       If you play with this program please send changes
C       to (starting at 02/01/96)
C         bauer@cvtserv1.verfahrenstechnik.uni-stuttgart.de
C       before send changes to
C         bauer@ra.me.wisc.edu
C
C       Thanks a lot
C           Matthias Bauer
C
C
C-----------------------------------------------------------------
C-----------------------------------------------------------------
c-----------------------------------------------------------------

      PROGRAM GRNN_MAIN
C_____________________________________________

C  DECLARATION OF VARIABLES
      double precision x_train,y_train   ! sample data
      double precision sigma_use,sigma_old ! smoothness parm
      double precision sigma_step        ! smoothness parm changes
      double precision X,Y,x_norm        ! evaluation points
      double precision maxi,mini         ! dummies to calculate max
and min

      double precision grnn_val          ! procedure value of GRNN

      integer n                          ! numb of training samples
      integer q                          ! numb of dimensions
      integer i,j,k                      ! counting vars
      integer numb_sigma                 ! counting vars
      integer inflections                ! numb of inflections allowed
      integer overshoot                  ! how many times over allowed
infl
      integer blocks                     ! blocks of data
      integer set_size                   ! size of block
```

```
      integer m                          ! numb of points in evaluat
file

C   Declaration of Dimensions
      dimension x_train(10000,10),y_train(10000)
      dimension X(20000,10),Y(20000),x_norm(20000,10)
      dimension maxi(10),mini(10)

      character*3 enough                 ! flag for iteration
      character*20  out_name, train_name, eval_name     ! filenames
      character*1   jump                 ! jump over iterations flag
      external readtrain                 ! procedure name
      external grnn                      ! procedure name

C_____________________________________________________
C
C       BEGINNING OF THE PROGRAM
C
C_____________________________________________________

c  reading trainng data
      write(*,*) 'trainnname?'
      write(*,*) 'make sure that training data are normalized!'
      read(*,*) train_name

      OPEN(333,file=train_name,status='unknown')
      rewind 333
      CALL READTRAIN(X_train,Y_train,n,q,numb_sigma,
     &          sigma_step,maxi,mini,inflections,blocks)
      CLOSE (333)

         write(*,*) 'evaluate name?'
         write(*,*) 'make sure that evaluation file is normalized!'
         read(*,999) eval_name

c----- reading evaluating data
      OPEN(444,file=eval_name,status='unknown')
        rewind 444
        READ(444,*)
        READ(444,*)
        READ(444,*) M             ! number of evaluation points
        READ(444,*) Q             ! dimenisions of input vector
        READ(444,*) set_size      ! how many samples for each change
in dim.
        READ(444,*) sigma_use     ! can be used for input of sigma,
then
        write(*,*) 'what start sigma?'  ! comment those out!
        READ(*,*) sigma_use       ! sigma start for wiggle method
        DO i=1,m
           READ(444,*) (X_norm(i,j), j=1,Q)   ! eval points
        END DO
      CLOSE (444)
```

```
c  reading output file name!
         write(*,*) 'filename output?'
         read(*,999) out_name

C----------------------------------------
C GRNN
C----------------------------------------
         jump = 'no'
         j=0
c  allow possibility to circumvent wiggle method
         write(*,*) 'do you want to use a specific sigma? y/n'
         read(*,*) jump
         if ( jump .eq. 'y') then
            write(*,*) 'what sigma would you like to use?'
            read(*,*) sigma_use
         end if
c  loop iteration for wiggle method
         do while (enough .ne. 'yes')
            j=j+1
c cushen possible wrong (too low) inflectionpoints
         if (j .gt. 100) then
             inflections = inflections +1
             j = 0
             sigma_use = 0.005   ! st back of sigma
             write(*,*) 'inflections were increased to: ',inflections
             read(*,*)    ! empty line to confirm increase iof infl.
          end if

c  jump to subroutine GRNN for all trainng samples (Neural Net here!)
          DO i=1,m
            call GRNN(N,Q,i,sigma_use,
     &            X_norm,X_train,Y_train,grnn_val
     &            ,out_name)
            y(i) = grnn_val
          END DO

          sigma_old = sigma_use   ! store old sigma

c------  call of subroutine to get optimal sigma
c        using inflection points
          if (jump .ne. 'y' ) then
             write(*,*) 'before curve',inflections
              call curve(y,n,sigma_use,overshoot,enough,
     &              inflections,set_size,q)
          end if
          write(*,*) 'after curve'
          if (jump .eq. 'y') then
             enough = 'yes'
          end if

        end do
```

```
        sigma_use=sigma_old

c-------------------------------------------
c   output
c-------------------------------------------

c ------- output in file
      OPEN(444,file=out_name,status='unknown')
        rewind 444
        write(444,888) M
        write(444,888) Q
        write(444,777) sigma_use
        DO i=1,M
           write(444,777) (X(i,j), j=1,Q),Y(i)
        END DO
      CLOSE (444)

c poosibility for two dim prob for easier plot in excel or elsewhere!
      if (q .eq. 2) then
         write(*,*) 'outname for special file?'
         read(*,*) out_name
c ------- output in file special for 2 input dimensions
      OPEN(444,file=out_name,status='unknown')
        rewind 444
        DO i=1,m,set_size
           write(444,779) (Y(i+k),k=0,set_size-1)
        END DO
      end if
      CLOSE (444)
      close(999)

999     format(A20)
888     format(I8)
777     format(10F15.8)
778     format(10F15.4)
779   format(51F15.4)

      END

C_____________________________________________________________
C
C           BELOW HERE: SUBROUTINES AND FUNCTIONS
C
C_____________________________________________________________

C-----------------------------------------------
C   SUBROUTINE GRNN
C-----------------------------------

      subroutine GRNN(N,Q,i_main,
     &            sigma,X,X_train,Y_train,grnn_val
     &            ,out_name)
```

```
      double precision sigma        ! sigma
      double precision DENOM,NUM    ! neuron values
      double precision D2           ! distance^2
      double precision X            ! evaluation vector
      double precision X_train,Y_train ! training samples
      double precision grnn_val     ! predicted result
      double precision dummy,dummy2 ! dummies

      integer i,j    ! counting dimensions in array
      integer n      ! max numb in array
      integer q      ! dimenison of vector
      integer i_main ! evaluation sample

      character*20 out_name

c dimensions of trainng samples
      dimension x_train(10000,10),Y_train(10000)
      dimension X(20000,10)

C-----------------------------

c reset values of neurons
      NUM =0.0
      DENOM=0.0
      d2=0.0

c core of GRNN
      do i=1,n
c distance
         d2=0.0
         do j   = 1,q
            D2  = D2 + (X_train(i,j)-X(i_main,j))**2.0
         end do
         dummy2 = -D2/(2.0*sigma**2.0)
         dummy  = DEXP(dummy2)
         NUM    = NUM+Y_train(i)*dummy
         DENOM  = DENOM+dummy
      END DO

c cushen numerical problems
      if (DENOM .le. 1.0D-140) Then
         grnn_val = 1.0D140
c        write(*,*) 'value reset!',out_name,denom
      else
         grnn_val = NUM/DENOM
      end if

      RETURN
      END

c-------------------------------------------------------
```

```
c  WIGGLE METHOD
c-------------------------------------------------------

      subroutine curve(y,n,sigma,overshoot,enough,countmax,
     &                 set_size,q)

      double precision y          ! predicted values
      double precision sigma      ! smoothness parm
      double precision sec1,sec2  ! second derivatives
      double precision dummy      ! dummy variable

      integer n            ! numb of eval. points
      integer overshoot    ! overshooting over wanted inflections
      integer i,jj,kkk     ! counting vars
      integer countmax,count ! counting inflections
      integer j,set_size,q   ! data charac. for wiggle method
      integer max_count,ave_count  ! results in inflection count

      character*3 enough

      dimension y(20000)

      write(*,*) set_size

      enough = 'no'
      max_count=0
      ave_count=0

c-------------for only one dimension---------------

      if (q .eq. 1) then
         count = 0
         sec1= y(1)-2.0*y(2)+y(3)
        do j=3,set_size-1
         sec2 = y(j-1)-2.0*y(j)+y(j+1)
         dummy = sec1*sec2
         if (dummy .le. 0.0 ) then
            if ( (sec1 .eq. 0.0 )
     &             .and. (sec2 .eq. 0.0 ) ) then
             count=count
            else
             count= count+1
            end if
         end if
        sec1=sec2
        end do
        if (count .gt. max_count) then
           max_count =count
        end if
        ave_count =ave_count+count
        ave_denom=ave_denom+1
        end if
```

```
c--------------for two dimesnions--------------


c--------------------------------------
c derivative in first dimension
c--------------------------------------

      if (q .eq. 2) then
      count=0
c in first dimension!
      do i=1,set_size
         sec1 = y((i-1)*set_size+1)
     &       -2.0*y((i-1)*set_size+2)
     &           +y((i-1)*set_size+3)
        do j=3,set_size-1
        ii=ii+1
         sec2 = y((i-1)*set_size+j-1)
     &       -2.0*y((i-1)*set_size+j)
     &           +y((i-1)*set_size+j+1)
         dummy = sec1*sec2
         if (dummy .le. 0.0 ) then
            if ( (sec1 .eq. 0.0 )
     &             .and. (sec2 .eq. 0.0 ) ) then
             count=count
            else
c            count= count+1
            end if
         end if
       sec1=sec2
       end do
       if (count .gt. max_count) then
c         max_count =count
       end if
       ave_count =ave_count+count
       ave_denom=ave_denom+1
       count=0
      end do

c--------------------------------------
c derivative in second dimension
c--------------------------------------

      count=0
      do i=1,set_size
         sec1 = y((i-1)+1)
     &       -2.0*y((i-1)+1  +set_size)
     &           +y((i-1)+1+2*set_size)
        do j=3,set_size-1
         sec2 = y((i-1)+set_size*(j-1))
     &       -2.0*y((i-1)+set_size*j)
     &           +y((i-1)+set_size*(j+1))
```

```
          dummy = sec1*sec2
          if (dummy .le. 0.0 ) then
             if ( (sec1 .eq. 0.0 )
     &             .and. (sec2 .eq. 0.0 ) ) then
              count=count
             else
              count= count+1
             end if
          end if
         sec1=sec2
         end do
         if (count .gt. max_count) then
            max_count =count
         end if
         ave_count =ave_count+count
         ave_denom=ave_denom+1
         count=0
        end do

        end if

c------------three dimensions----------------------

c--------------------------------------
c derivative in first dimension
c--------------------------------------

        if (q .eq. 3) then
        count=0
        do jj=1,set_size
         do i=1,set_size
           sec1 = y(1+  (i-1)*set_size+(jj-1)*set_size**2)
     &       -2.0*y(2+  (i-1)*set_size+(jj-1)*set_size**2)
     &           +y(3+  (i-1)*set_size+(jj-1)*set_size**2)
          do j=3,set_size-1
           sec2 = y(j-1+(i-1)*set_size+(jj-1)*set_size**2)
     &       -2.0*y(j+  (i-1)*set_size+(jj-1)*set_size**2)
     &           +y(j+1+(i-1)*set_size+(jj-1)*set_size**2)
           dummy = sec1*sec2
           if (dummy .le. 0.0 ) then
              if ( (sec1 .eq. 0.0 )
     &             .and. (sec2 .eq. 0.0 ) ) then
              count=count
             else
              count= count+1
             end if
           end if
         sec1=sec2
         end do
         if (count .gt. max_count) then
            max_count =count
         end if
```

```
         ave_count =ave_count+count
         ave_denom=ave_denom+1
         count=0
         end do
        end do

    c-------------------------------------
    c derivative in second dimension
    c-------------------------------------

        count=0
        do jj=1,set_size
         do i=1,set_size
           sec1 = y(i+0*set_size+(jj-1)*set_size**2)
         &       -2.0*y(i+1*set_size+(jj-1)*set_size**2)
         &           +y(i+2*set_size+(jj-1)*set_size**2)
           do j=3,set_size-1
            sec2 = y(i+(j-1)*set_size+(jj-1)*set_size**2)
         &       -2.0*y(i+(j  )*set_size+(jj-1)*set_size**2)
         &           +y(i+(j+1)*set_size+(jj-1)*set_size**2)
            dummy = sec1*sec2
            if (dummy .le. 0.0 ) then
               if ( (sec1 .eq. 0.0 )
         &             .and. (sec2 .eq. 0.0 ) ) then
                count=count
               else
                count= count+1
               end if
           end if
          sec1=sec2
          end do
          if (count .gt. max_count) then
             max_count =count
          end if
          ave_count =ave_count+count
          ave_denom=ave_denom+1
          count=0
          end do
        end do

    c-------------------------------------
    c derivative in third dimension
    c-------------------------------------

        count=0
        do j=1,set_size
         do i=1,set_size
             sec1 =
         &             y(i+(j-1)*set_size+0*(set_size)**2)
         &        -2.0*y(i+(j-1)*set_size+1*(set_size)**2)
         &            +y(i+(j-1)*set_size+2*(set_size)**2)
            do jj=3,set_size-1
```

```
            ii=ii+1
            sec2 =
     &             y(i+(j-1)*set_size+(jj-1)*(set_size)**2)
     &        -2.0*y(i+(j-1)*set_size+(jj  )*(set_size)**2)
     &            +y(i+(j-1)*set_size+(jj+1)*(set_size)**2)
          dummy = sec1*sec2
          if (dummy .le. 0.0 ) then
             if ( (sec1 .eq. 0.0 )
     &             .and. (sec2 .eq. 0.0 ) ) then
           count=count
          else
           count= count+1
          end if
        end if
        sec1=sec2
        end do
        if (count .gt. max_count) then
           max_count =count
        end if
        ave_count =ave_count+count
        ave_denom=ave_denom+1
        count=0
        end do
       end do

       end if

c--------------------dimension = 4 -----------------

       if (q .eq. 4) then

c---------------------------------------
c   derivative in first dimension
c---------------------------------------
       count=0
       do kkk=1,set_size
        do jj=1,set_size
         do i=1,set_size
          sec1 = y(1+  (i-1)*set_size+(jj-1)*set_size**2
     &                           +(kkk-1)*set_size**3)
     &       -2.0*y(2+  (i-1)*set_size+(jj-1)*set_size**2
     &                           +(kkk-1)*set_size**3)
     &           +y(3+  (i-1)*set_size+(jj-1)*set_size**2
     &                           +(kkk-1)*set_size**3)
        do j=3,set_size-1
          sec2 = y(j-1+(i-1)*set_size+(jj-1)*set_size**2
     &                           +(kkk-1)*set_size**3)
     &       -2.0*y(j+  (i-1)*set_size+(jj-1)*set_size**2
     &                           +(kkk-1)*set_size**3)
     &           +y(j+1+(i-1)*set_size+(jj-1)*set_size**2
     &                           +(kkk-1)*set_size**3)
          dummy = sec1*sec2
```

```
            if (dummy .le. 0.0 ) then
               if ( (sec1 .eq. 0.0 )
     &              .and. (sec2 .eq. 0.0 ) ) then
                count=count
               else
                count= count+1
               end if
            end if
           sec1=sec2
           end do
           if (count .gt. max_count) then
              max_count =count
           end if
           ave_count =ave_count+count
           ave_denom=ave_denom+1
           count=0
           end do
         end do
        end do


c---------------------------------------
c   derivative in second dimension
c---------------------------------------
        count=0
        do kkk=1,set_size
        do jj=1,set_size
         do i=1,set_size
           sec1 = y(i+0*set_size+(jj-1)*set_size**2
     &                       +(kkk-1)*set_size**3)
     &       -2.0*y(i+1*set_size+(jj-1)*set_size**2
     &                       +(kkk-1)*set_size**3)
     &           +y(i+2*set_size+(jj-1)*set_size**2
     &                       +(kkk-1)*set_size**3)
           do j=3,set_size-1
            sec2 = y(i+(j-1)*set_size+(jj-1)*set_size**2
     &                       +(kkk-1)*set_size**3)
     &       -2.0*y(i+(j  )*set_size+(jj-1)*set_size**2
     &                       +(kkk-1)*set_size**3)
     &           +y(i+(j+1)*set_size+(jj-1)*set_size**2
     &                       +(kkk-1)*set_size**3)
            dummy = sec1*sec2
            if (dummy .le. 0.0 ) then
               if ( (sec1 .eq. 0.0 )
     &              .and. (sec2 .eq. 0.0 ) ) then
                count=count
               else
                count= count+1
               end if
           end if
          sec1=sec2
          end do
```

```
       if (count .gt. max_count) then
          max_count =count
       end if
       ave_count =ave_count+count
       ave_denom=ave_denom+1
       count=0
       end do
      end do
      end do

c-------------------------------------
c derivative in third dimension
c-------------------------------------
      count=0
      do kkk=1,set_size
      do j=1,set_size
       do i=1,set_size
           sec1 =
     &             y(i+(j-1)*set_size+0*(set_size)**2
     &                           +(kkk-1)*set_size**3)
     &        -2.0*y(i+(j-1)*set_size+1*(set_size)**2
     &                           +(kkk-1)*set_size**3)
     &            +y(i+(j-1)*set_size+2*(set_size)**2
     &                           +(kkk-1)*set_size**3)
         do jj=3,set_size-1
           ii=ii+1
           sec2 =
     &             y(i+(j-1)*set_size+(jj-1)*(set_size)**2
     &                           +(kkk-1)*set_size**3)
     &        -2.0*y(i+(j-1)*set_size+(jj  )*(set_size)**2
     &                           +(kkk-1)*set_size**3)
     &            +y(i+(j-1)*set_size+(jj+1)*(set_size)**2
     &                           +(kkk-1)*set_size**3)
         dummy = sec1*sec2
         if (dummy .le. 0.0 ) then
            if ( (sec1 .eq. 0.0 )
     &             .and. (sec2 .eq. 0.0 ) ) then
           count=count
           else
            count= count+1
           end if
         end if
        sec1=sec2
        end do
        if (count .gt. max_count) then
           max_count =count
        end if
        ave_count =ave_count+count
        ave_denom=ave_denom+1
        count=0
        end do
       end do
```

```
      end do


c-------------------------------------
c derivative in fourth dimension
c-------------------------------------
      count=0
      do jj=1,set_size
      do j=1,set_size
       do i=1,set_size
           sec1 =
     &             y(i+(j-1)*set_size+0*(set_size)**2
     &                                  +0*set_size**3)
     &        -2.0*y(i+(j-1)*set_size+1*(set_size)**2
     &                                  +1*set_size**3)
     &            +y(i+(j-1)*set_size+2*(set_size)**2
     &                                  +2*set_size**3)
         do kkk=3,set_size-1
           ii=ii+1
           sec2 =
     &             y(i+(j-1)*set_size+(jj-1)*(set_size)**2
     &                                  +(kkk-1)*set_size**3)
     &        -2.0*y(i+(j-1)*set_size+(jj  )*(set_size)**2
     &                                  +(kkk-1)*set_size**3)
     &            +y(i+(j-1)*set_size+(jj+1)*(set_size)**2
     &                                  +(kkk-1)*set_size**3)
         dummy = sec1*sec2
         if (dummy .le. 0.0 ) then
            if ( (sec1 .eq. 0.0 )
     &             .and. (sec2 .eq. 0.0 ) ) then
           count=count
           else
            count= count+1
           end if
         end if
        sec1=sec2
        end do
        if (count .gt. max_count) then
           max_count =count
        end if
        ave_count =ave_count+count
        ave_denom=ave_denom+1
        count=0
        end do
       end do
       end do

       end if

c---------change in sigma --------------------

       count = max_count
```

```
c     count = ave_count/ave_denom

      write(*,*) 'countmax',countmax,count

      if ( count .gt. countmax) then
         sigma = sigma*1.1
      else
         overshoot = overshoot +1
         if (overshoot .ge. 2) then
            sigma = sigma*0.93
            if (overshoot .gt. 8) then
             enough = 'yes'
            end if
         end if
      end if

c--------output on screen to check
      write(*,*) 'inflections counted, over goal'
      write(*,*) count,overshoot
      write(*,*) sigma

777     format(1F15.5)
888     format(I5)
      return
      end


c-------------------------------------------------------
C   SUBROUTINE TO READ TRAINING DATA
c-------------------------------------------------------

      SUBROUTINE READTRAIN(x_train,y_train,N,Q,numb_sigma,
     &        sigma_step,maxi,mini,countmax,blocks)

      double precision X_train,Y_train
      double precision sigma_step
      double precision maxi,mini

      character*20 cluster_flag

      INTEGER i , j , N, Q
      integer numb_sigma
      integer countmax         ! number of inflection points in train
      integer blocks

      DIMENSION X_train(10000,10), Y_train(10000)
      DIMENSION maxi(10),mini(10)

      write(*,*) 'reading data'

      READ(333,*)
      READ(333,*) cluster_flag
```

```
      write(*,*) 'how many samples?'
      READ(*,*) N
      READ(333,*) Q
      READ(333,*)
      READ(333,*)
      DO i=1,n
         READ(333,*) (X_train(i,j), j=1,Q),Y_train(i)
      END DO

c sample output of trainng data
      DO i=1,12
         write(*,*) (X_train(i,j), j=1,Q),Y_train(i)
      END DO

      write(*,*) ' how many inflection points do you allow?'
      read(*,*) countmax

      RETURN

999     format(A20)
888     format(I5)
777     format(10E16.7)

      END
```