# *Appendix D – Programs*

## Ret.f

Program usage: Ret.exe eaflag gflag nproc

eaflag=[1|2|3|4]
Method for computing ea (actual vapor press.):
eaflag = 1  Based on Tdew
eaflag = 2  Assuming Tdew=Tmin
eaflag = 3  Based on RHmax, RHmin, Tmax, Tmin
eaflag = 4  Input daily-average VPD (es-ea)

gflag=[0|1]
Method for computing G (ground heat flux):
gflag = 0  Neglected
gflag = 1  Based on daily air temp. change

nproc>=1
Number of stations to process

```
      PROGRAM REFET
*     ****************************************************************
*     Program to compute Daily Reference Grass ET (ETo) based on
*       1 - PENMAN MONTEITH METHOD as defined by FAO Irrigation and
*           Drainage Paper 56 (FAO-56)
*       2 - PRIESTLEY-TAYLOR METHOD with monthly-varying alpha coef.
*
*     Wet marsh potential ET (PET) by the Simple Method is also
*         computed
*     ****************************************************************
*     Input requirements:
*             rs  = downward solar radiation (MJ/m^2/day)
*            tmax = daily maximum temperature (C)
*            tmin = daily minimum temperature (C)
*
*             dew = daily average dew point temperature (C) OR
*           [rhmax = daily maximum relative humidity (%) AND
*            rhmin = daily minimum relative humidity (%)] OR
*              vpd = vapor pressure deficit (kPa)
*                    See eaflag below
*
*              uz = daily average wind speed (m/s)
*               p = daily average barometric pressure (kPa)
*              zm = wind measurement height (m)
*              zh = temperature/humidity measurement height (m)
*            glat = station latitude (deg)
*            elev = station elevation (m)
*          alphac = monthly alpha coefficients for Priestley-Taylor
*
*     Output:
*           etopm = ETo by Penman-Monteith (in/day)
*           etopt = ETo by Priestley-Taylor (in/day)
*
*         petsmpl = PET by Simple Method (in/day)
*     ****************************************************************
*
*     ***********************DEFINITIONS*****************************
      integer     nsta,nproc
      parameter   (nsta=400)

      integer     date,i,icount,iyear,imon,iday,ista,j,l

      real        pi
      parameter   (pi=3.141592654)
```

```
      real        acoeff
      real        alphac(12)

      real        hc,rl,alpha,cs,ds,gsc,as,bs,zcoeff,ac,bc,
     a            al,bl,vk

      real        tmax_min,ra,rc,es,ea,delta,lambda,gamma,
     a            gammod,rho,g

      real        dr,del,phi,ws,rad,rns,rso,fcl,epsi,rnl,rn

      real        zm,zh
      real        glat(nsta),elev(nsta)
      real        rs(nsta),tmax(nsta),tmin(nsta),dew(nsta),
     a            rhmax(nsta),rhmin(nsta),vpd(nsta),uz(nsta),
     a            p(nsta)
      real        t(nsta),tn(nsta),etrad(nsta),etaero(nsta),
     a            etopm(nsta),etopt(nsta),petsmpl(nsta)

      data        icount /0/

     character*50 cdum, ndum

      character*1 eaflag,rhflag,gflag

*     ***********************PARAMETERS*****************************

c     Reference crop height (m)
     hc = 0.12

c     Stomatal resistance of a single leaf (s/m)
     rl = 100.0

c     Albedo of reference crop
     alpha=0.23

c     Soil heat capacity (MJ/m^3/C)
     cs = 2.1

c     Effective soil depth (m) for daily temp. fluctuations
     ds = 0.18

c     Solar constant (MJ/m^2/min)
      gsc = 0.0820

c     Angstrom values (as recommended by Doorenbos & Pruitt, 1977)
c        if no calibrated values are available
     as = 0.25
     bs = 0.50

c     Elevation coefficient for clear-sky solar rad. (zcoeff, 1/m)
      zcoeff = 2E-5

c     Coefficients to compute cloudiness factor
     ac = 1.35
     bc = -0.35

c     Coefficients to compute net emissivity
     al = 0.34
     bl = -0.14

c     Von Karman's constant
      vk = 0.41

*     ****************************************************************

c     Check number of arguments passed to program
      if (iargc().lt.4) then
         write (*,*) ''
         write (*,*) 'Only', iargc(), ' arguments have been entered'
         write (*,*) ''
         write (*,*) 'Program usage:'
         write (*,*) 'Ret.exe eaflag rhflag gflag nproc'
         write (*,*) ''
         write (*,*) 'eaflag=[1|2|3|4]'
```

```
            write (*,*) 'Method for computing ea (actual vapor press.):'
            write (*,*) 'eaflag = 1  Based on Tdew'
            write (*,*) 'eaflag = 2  Assuming Tdew=Tmin'
            write (*,*) 'eaflag = 3  Based on RHmax, RHmin, Tmax, Tmin'
            write (*,*) 'eaflag = 4  Input computed VPD (es-ea)'
            write (*,*) ''
            write (*,*) 'rhflag=[0|1]'
            write (*,*) 'Flag to cap relative humidity to 100%'
            write (*,*) 'Only applies if eaflag = 3'
            write (*,*) 'rhflag = 0  Do not cap RH'
            write (*,*) 'rhflag = 1  Cap RH'
            write (*,*) ''
            write (*,*) 'gflag=[0|1]'
            write (*,*) 'Method for computing G (ground heat flux):'
            write (*,*) 'gflag = 0  Neglected'
            write (*,*) 'gflag = 1  Based on daily air temp. change'
            write (*,*) ''
            write (*,*) 'nproc>=1'
            write (*,*) 'Number of stations to process'
            write (*,*) ''
            stop
         endif

         call getarg(1,eaflag)
         call getarg(2,rhflag)
         call getarg(3,gflag)
         call getarg(4,ndum)

         read (unit=ndum,fmt='(i5)') nproc
         if (nproc.lt.1) then
            write (*,*) 'Number of stations to process must be >= 1'
            stop
         endif

         if (nproc.gt.nsta) then
            write (*,*) 'Number of stations to process:', nproc
            write (*,*) 'is larger than array dimensions:', nsta
            write (*,*) ''
            write (*,*) 'Modify parameter nsta in program '
            write (*,*) 'to increase array size'
            stop
         endif

*        ************************************************************

c        Open files
         open (1, file = 'alpha_coeff_PT.dat', access ='sequential',
     a  form ='formatted',status ='old')

         open (2, file = 'dswrf.txt', access ='sequential',
     a  form ='formatted',status ='old')

         open (3, file = 'tmax.txt', access ='sequential',
     a  form ='formatted',status ='old')

         open (4, file = 'tmin.txt', access ='sequential',
     a  form ='formatted',status ='old')

         if (eaflag.eq.'1') then
            open (5, file = 'tdew.txt', access ='sequential',
     a              form ='formatted',status ='old')
         else if (eaflag.eq.'3') then
            open (5, file = 'rhmax.txt', access ='sequential',
     a              form ='formatted',status ='old')

            open (6, file = 'rhmin.txt', access ='sequential',
     a              form ='formatted',status ='old')
         else if (eaflag.eq.'4') then
            open (5, file = 'vpd.txt', access ='sequential',
     a              form ='formatted',status ='old')
         endif

         open (7, file = 'wind.txt', access ='sequential',
     a  form ='formatted',status ='old')
```

```
        open (8, file = 'pres.txt', access ='sequential',
     a  form ='formatted',status ='old')

        open (9, file = 'dataset_metadata.txt', access ='sequential',
     a  form ='formatted',status ='old')

        open (10, file = 'station_metadata.txt', access ='sequential',
     a  form ='formatted',status ='old')

        open (21, file = 'etopm.txt', access ='sequential',
     a  form ='formatted',status ='unknown')

        open (22, file = 'etopt.txt', access ='sequential',
     a  form ='formatted',status ='unknown')

        open (23, file = 'petsmpl.txt', access ='sequential',
     a  form ='formatted',status ='unknown')

*       ************************************************************

c       Read in alpha coefficients for Priestley-Taylor
        read (1,*) cdum

 100    continue
           read (1, *, end=200) imon, acoeff
           alphac(imon) = acoeff
           go to 100
 200    continue

c       Read in dataset metadata (zm, zh)
       read (9, *) zm, zh

c       Read in station metadata (glat, elev)
        ista = 1
 150    continue
          read (10, *, end=250) glat(ista), elev(ista)
           ista = ista + 1
           go to 150
 250    continue
        if (ista-1.lt.nproc) then
           write (*,*) 'Missing station metadata'
           write (*,*) 'data for',nproc,' stations expected'
           write (*,*) 'data only found for',ista-1,' stations'
           stop
        endif

c       Read in date, rs, tmax, tmin, rhmax, rhmin, uz, p
 300    continue
           read (2, *, end=400) date, (rs(i),i=1,nproc)
c           write (*,*) date, ' rs', rs(1)
           read (3, *, end=400) date, (tmax(i),i=1,nproc)
c           write (*,*) date, 'tmax', tmax(1)
           read (4, *, end=400) date, (tmin(i),i=1,nproc)
c           write (*,*) date, 'tmin', tmin(1)

           if (eaflag.eq.'1') then
              read (5, *, end=400) date, (dew(i),i=1,nproc)
c              write (*,*) date, 'dew', dew(1)
           else if (eaflag.eq.'3') then
              read (5, *, end=400) date, (rhmax(i),i=1,nproc)
c              write (*,*) date, 'rhmax', rhmax(1)
              read (6, *, end=400) date, (rhmin(i),i=1,nproc)
c              write (*,*) date, 'rhmin', rhmin(1)
           else if (eaflag.eq.'4') then
              read (5, *, end=400) date, (vpd(i),i=1,nproc)
c              write (*,*) date, 'vpd', vpd(1)
           endif

           read (7, *, end=400) date, (uz(i),i=1,nproc)
c           write (*,*) date, 'uz', uz(1)
           read (8, *, end=400) date, (p(i),i=1,nproc)
c           write (*,*) date, 'p', p(1)

c          Increment icount
           icount = icount+1
```

```
c          Extract year, month, day from date
           iyear = date/10000
           imon = (date - iyear*10000)/100
           iday = (date - iyear*10000) - imon*100

c          Determine whether year is a leap year
c             Years divisible by 400 are leap years
           if (mod(iyear,400).eq.0) then
              l = 1
c             Other centuries are not leap years
           else if (mod(iyear,100).eq.0) then
              l = 0
c             Otherwise, every fourth year is a leap year
           else if (mod(iyear,4).eq.0) then
              l = 1
c             Other years are not leap years
           else
              l = 0
           endif

c          Compute julian day (Annex 2 FAO-56)
          j = int(275.0*float(imon)/9.0-30.0+float(iday))-2

          if (float(imon).lt.3.0) then
                 j = j+2
          else if ((l.eq.1) .and. (float(imon).gt.2.0)) then
                 j = j+1
          endif
           write (*,*) 'Processing ',iyear,imon,iday,' (',j,' )'

c          Loop through all stations
           do 350 ista = 1,nproc

c             Compute daily average temp. and temp. range
              t(ista) = (tmax(ista)+tmin(ista))/2.
c              write (*,*) 't', t(ista)
              tmax_min = tmax(ista)-tmin(ista)

c             Compute aerodynamic resistance (ra, s/m)
              call aerores(vk,zm,hc,zh,uz(ista),ra)
c              write (*,*) 'ra', ra

c             Compute bulk canopy resistance (rc, s/m)
              call cropres(rl,hc,rc)
c              write (*,*) 'rc', rc

c             Compute saturated vapor pressure (es, kPa)
              call satvap(tmax(ista),tmin(ista),es)
c              write (*,*) 'es', es

c             Compute vapor pressure deficit
c             Method for computing ea:
c                eaflag = '1'  Based on Tdew,
c                eaflag = '2'  Assuming Tdew=Tmin
c                eaflag = '3'  Based on RHmax, RHmin, Tmax, Tmin
c                eaflag = '4'  Input computed VPD (es-ea)
              if (eaflag.ne.'4') then
c               Compute actual vapor pressure (ea, kPa)
                call actvap(eaflag,rhflag,tmax(ista),tmin(ista),
     a                      dew(ista),rhmax(ista),rhmin(ista),ea)
c                write (*,*) 'ea', ea

c               Compute vapor pressure deficit (vpd, kPa)
                call vpdef(es,ea,vpd(ista))
c                write (*,*) 'vpd', vpd(ista)
              else
c               Estimate actual vapor pressure (ea, kPa)
                ea = es - vpd(ista)
c                write (*,*) 'ea', ea
c                write (*,*) 'vpd', vpd(ista)
              endif

c             Compute slope of sat. vapor press. curve (delta, kPa/C)
              call slope(es,t(ista),delta)
```

```
c              write (*,*) 'delta', delta

c             Compute latent heat of vaporization (lambda, MJ/kg)
              call latheat(t(ista),lambda)
c              write (*,*) 'lambda', lambda

c             Compute psychrometric constant (gamma, kPa/C)
              call psyconst(p(ista),lambda,gamma)
c              write (*,*) 'gamma', gamma

c             Compute modified psychrometric constant (gammod, kPa/C)
              call modpsyconst(gamma,rc,ra,gammod)
c              write (*,*) 'gammod', gammod

c             Compute mean air density (rho, kg/m^3)
              call dense(p(ista),ea,t(ista),rho)
c              write (*,*) 'rho', rho

c             Compute soil heat flux (g, MJ/m^2/day)
c             Method for computing g:
c                gflag = '0'  Neglected
c                gflag = '1'  Based on daily air temperature change
              if (gflag.eq.'0') then
c             Soil heat flux is neglected
                 g = 0.0
              else
                 if (icount.eq.1) tn(ista)=t(ista)
                call gterm(cs,ds,t(ista),tn(ista),g)
              endif
c              write (*,*), 'g', g

c             Compute inverse distance Earth-Sun (dr)
              dr = 1.0+0.033*cos(2*pi/365*j)
c              write (*,*) 'dr', dr

c             Compute solar declination (del, rad)
              del = 0.409*sin(2*pi/365*j - 1.39)
c              write (*,*) 'del', del

c             Compute station latitude (phi, rad)
              phi = glat(ista)*pi/180.
c              write (*,*) 'phi', phi

c             Compute sunset hour angle (ws, rad)
              ws = acos(-tan(phi)*tan(del))
c              write (*,*) 'ws', ws

c             Compute extraterrestrial solar radiation
c             (rad, MJ/m^2/day)
              rad = 24*60/pi*gsc*dr*(ws*sin(phi)*sin(del)+
     a              cos(phi)*cos(del)*sin(ws))
c              write (*,*) 'rad', rad

c             Compute net solar radiation (rns, MJ/m^2/day)
              rns = (1.0-alpha)*rs(ista)
c              write (*,*) 'rns', rns

c             Compute clear-sky solar radiation (rso, MJ/m^2/day)
              rso = (as+bs+zcoeff*elev(ista))*rad
c              write (*,*) 'rso', rso, ' elev', elev(ista), as, bs,
c     a                     zcoeff, elev(ista)

c             Compute cloudiness factor (fcl)
              fcl = max(0.0,ac*min(1.0,(rs(ista)/rso)) + bc)
c              write (*,*) 'fcl', fcl

c             Compute net emissivity (epsi)
              epsi = (al + bl*sqrt(ea))
c              write (*,*) 'epsi', epsi

c             Compute net longwave radiation (rnl, MJ/m^2/day)
              rnl = 4.903e-9*(0.5*((tmax(ista)+273.16)**4+
     a               (tmin(ista)+273.16)**4))*epsi*fcl
c              write (*,*) 'rnl', rnl
```

```
c             Compute net radiation (rn, MJ/m^2/day)
              rn = rns - rnl
c              write (*,*) 'rn', rn

c             Compute radiation component of ETo by Penman-Monteith
c             (etrad, mm/day)
              etrad(ista) = delta*(rn-g)/((delta+gammod)*lambda)
c              write (*,*) 'etrad', etrad(ista)

c             Compute aerodynamic component of ETo by Penman-Monteith
c             (etaero, mm/day)
              etaero(ista) = 86.4*rho*622.
     a                *gamma*vpd(ista)/((delta+gammod)*ra*p(ista))
c              write (*,*) 'etaero', etaero(ista)

c             Compute ETo by Penman-Monteith (etopm, in/day)
c             25.4 is conversion from mm/day to in/day
              etopm(ista)  = (etrad(ista)+etaero(ista))/25.4
c              write (*,*) 'etopm', etopm(ista)

c             Compute ETo by Priestley-Taylor (etopt, in/day)
c             25.4 is conversion from mm/day to in/day
              etopt(ista) = alphac(float(imon))*delta*(rn-g)
     a                /((delta+gamma)*lambda)/25.4
c              write (*,*) 'etopt', etopt(ista)

c             Compute wet marsh PET by Simple Method (petsmpl, in/day)
c             25.4 is conversion from mm/day to in/day
              petsmpl(ista) = 0.53*rs(ista)/lambda/25.4
c              write (*,*) 'petsmpl', petsmpl(ista)

c             Save previous timestep temperature
              tn(ista) = t(ista)

 350       continue

c          Write out results
           write (21,50) iyear,imon,iday,j,(etopm(i),i=1,nproc)
           write (22,50) iyear,imon,iday,j,(etopt(i),i=1,nproc)
           write (23,50) iyear,imon,iday,j,(petsmpl(i),i=1,nproc)
 50        format (i4,3(2x,i3),2x,400(f7.5,2x))
           go to 300

 400    continue

        close (1)
        close (2)
        close (3)
        close (4)
        close (5)
        close (6)
        close (7)
        close (8)
        close (9)
        close (21)
        close (23)
        close (23)

        stop
        end


*       **********************SUBROUTINES*****************************

c       Compute aerodynamic resistance (ra, s/m)
        subroutine aerores(vk,zm,hc,zh,uz,ra)

          ra = log((zm-0.66*hc)/(0.123*hc))*log((zh-0.66*hc)/(0.0123*hc))
     a         /(vk**2*uz)
c         ra = log((zm-0.08)/(0.015))*log((zh-0.08)/(0.0015))
c     a         /(vk**2*uz)
c           write (*,*) 'inside ra',zm,hc,zh,vk,uz
        end

c       Compute bulk canopy resistance (rc, s/m)
```

```
      subroutine cropres(rl,hc,rc)

         ail = 24*hc

         rc = rl/(0.5*ail)
      end

c     Compute saturated vapor pressure (es, kPa)
      subroutine satvap(tmax,tmin,es)

         call vappr(tmax,estmax)
         call vappr(tmin,estmin)
         es = (estmax+estmin)/2.
      end

c     Compute actual vapor pressure (ea, kPa)
      subroutine actvap(eaflag,rhflag,tmax,tmin,dew,rhmax,rhmin,ea)
          character*1 eaflag,rhflag
         if (eaflag.eq.'1') then
            call vappr(dew,ea)
          else if (eaflag.eq.'2') then
            call vappr(tmin,ea)
         else if (eaflag.eq.'3') then
            call vappr(tmax,estmax)
            call vappr(tmin,estmin)
            if (rhflag.eq.'1') then
               rhmax=min(100.0,rhmax)
               rhmin=min(100.0,rhmin)
            endif
           ea = 0.5*(estmin*rhmax/100.+estmax*rhmin/100.)
        endif
      end

c     Compute vapor pressure (e, kPa)
      subroutine vappr(temp,e)

         e = 0.611*exp(17.27*temp/(temp+237.3))
      end

c     Compute vapor pressure deficit (vpd, kPa)
      subroutine vpdef(es,ea,vpd)

         vpd = es - ea
      end

c     Compute slope of saturation vapor pressure curve (delta, kPa/C)
      subroutine slope(es,t,delta)

         delta = 4098.0*es/(t+237.3)**2
      end

c     Compute latent heat of vaporization (lambda, MJ/kg)
      subroutine latheat(t,lambda)

          real lambda
         lambda = 2.501 - (0.002361*t)
c          write (*,*) 'inside lambda', t, lambda
      end

c     Compute psychrometric constant (gamma, kPa/C)
      subroutine psyconst(p,lambda,gamma)

          real lambda
         gamma = 0.00163*p/lambda
      end

c     Compute modified psychrometric constant (gammamod, kPa/C)
      subroutine modpsyconst(gamma,rc,ra,gammod)

         gammod = gamma*(1.0+rc/ra)
      end

c     Compute mean air density (rho, kg/m^3)
      subroutine dense(p,ea,t,rho)
```

```
         rho = 3.486*p*(1.-0.378*ea/p)/(t+273.16)
      end

c      Compute soil heat flux (g, MJ/m^2/day)
      subroutine gterm(cs,ds,t,tn,g)

         g = cs*ds*((t-tn)/1)
      end
```

## Stats.f

Program usage: Stats.exe fmt nproc

fmt=[1|2]
Data format:
fmt = 1:  yyyymmdd, data1, data2, ...
fmt = 2:  yyyy, mm, dd, jjj, data1, data2, ...

nproc>=1'
Number of stations to process'

Use fmt = 1 to get statistics of meteorological data produced by Python program.
Use fmt = 2 to get statistics of RET and PET computed by FORTRAN Ret program.

Input file should be called data.txt.

```
      PROGRAM STATS
*     ****************************************************************
*     Program to compute mean ave, average deviation adev,
*       standard deviation sdev, variance var, skewness skew,
*       kurtosis kurt, and coefficient of variation cv.
*     ****************************************************************

*     **********************DEFINITIONS*****************************
      integer      nsta,nm,nproc,ny,nydata
      parameter    (nsta=400,nm=12,ny=200)

      integer      date,i,icount,iyear,imon,iday,iyrprev,ista,j,l
      integer      ctm(nm),cty(ny)
      integer      years(ny)

      real         avecty
      real         s(nm,nsta),p(nm,nsta)
      real         dat(nsta),ave(nm,nsta),adev(nm,nsta),sdev(nm,nsta),
     a             var(nm,nsta),skew(nm,nsta),kurt(nm,nsta),
     a             cv(nm,nsta)
      real         sy(ny,nsta)
      real         sa(nsta),avea(nsta),sdeva(nsta),cva(nsta)

      data         icount/0/,iyrprev/0/,nydata/0/
      data         ctm/nm*0/,cty/ny*0/
      data         sa/nsta*0.0/,avea/nsta*0.0/,sdeva/nsta*0.0/,
     a             cva/nsta*0.0/
      character*1  aveflag,fmt
      character*50 ndum
*     ****************************************************************

c     Check number of arguments passed to program
      if (iargc().lt.3) then
         write (*,*) ''
         write (*,*) 'Only', iargc(), ' arguments have been entered'
         write (*,*) ''
         write (*,*) 'Program usage: Stats.exe aveflag fmt nproc'
         write (*,*) ''
         write (*,*) 'aveflag=[0|1]'
         write (*,*) 'Flag for annual statistics'
         write (*,*) 'aveflag = 0:  Compute annual total statistics'
         write (*,*) 'aveflag = 1:  Compute annual average statistics'
         write (*,*) ''
         write (*,*) 'fmt=[1|2]'
         write (*,*) 'Data format:'
         write (*,*) 'fmt = 1:  yyyymmdd, data1, data2, ...'
         write (*,*) 'fmt = 2:  yyyy, mm, dd, jjj, data1, data2, ...'
         write (*,*) ''
```

```
            write (*,*) 'nproc>=1'
            write (*,*) 'Number of stations to process'
            write (*,*) ''
            stop
         endif

         call getarg(1,aveflag)
         call getarg(2,fmt)
         call getarg(3,ndum)

         read (unit=ndum,fmt='(i5)') nproc
         if (nproc.lt.1) then
            write (*,*) 'Number of stations to process must be >= 1'
            stop
         endif

         if (nproc.gt.nsta) then
            write (*,*) 'Number of stations to process:', nproc
            write (*,*) 'is larger than array dimensions:', nsta
            write (*,*) ''
            write (*,*) 'Modify parameter nsta in program '
            write (*,*) 'to increase array size'
            stop
         endif

*        ***************************************************************

c        Open files
        open (1, file = 'data.txt', access ='sequential',
     a  form ='formatted',status ='old')

c        Climatology files
         open (11, file = 'ave_clim.txt', access ='sequential',
     a   form ='formatted',status ='unknown')

         open (12, file = 'adev_clim.txt', access ='sequential',
     a   form ='formatted',status ='unknown')

         open (13, file = 'sdev_clim.txt', access ='sequential',
     a   form ='formatted',status ='unknown')

         open (14, file = 'var_clim.txt', access ='sequential',
     a   form ='formatted',status ='unknown')

         open (15, file = 'skew_clim.txt', access ='sequential',
     a   form ='formatted',status ='unknown')

         open (16, file = 'kurtexc_clim.txt', access ='sequential',
     a   form ='formatted',status ='unknown')

         open (17, file = 'cv_clim.txt', access ='sequential',
     a   form ='formatted',status ='unknown')

c        Files with annual statistics
         if (aveflag.eq.'0') then
            open (18, file = 'annual_totals.txt', access ='sequential',
     a      form ='formatted',status ='unknown')
         else
            open (18, file = 'annual_ave.txt', access ='sequential',
     a      form ='formatted',status ='unknown')
         endif

         open (19, file = 'annual_stats.txt', access ='sequential',
     a   form ='formatted',status ='unknown')
*        ***************************************************************

c        Initialize variabes
         do imon=1,nm
           do ista=1,nproc
             s(imon,ista)=0.0
             p(imon,ista)=0.0
             ave(imon,ista)=0.0
             adev(imon,ista)=0.0
             sdev(imon,ista)=0.0
             var(imon,ista)=0.0
```

```
            skew(imon,ista)=0.0
            kurt(imon,ista)=0.0
            cv(imon,ista)=0.0
          end do
        end do

        do iyear=1,ny
          do ista=1,nproc
            sy(iyear,ista)=0.0
          end do
        end do

c       Read-in data to compute means
 300    continue
        if (fmt.eq.'1') then

           read (1, *, end=400) date,(dat(i),i=1,nproc)

c          Extract year, month, day from date
           iyear = date/10000
           imon = (date - iyear*10000)/100
           iday = (date - iyear*10000) - imon*100

c          Determine whether year is a leap year
c             Years divisible by 400 are leap years
           if (mod(iyear,400).eq.0) then
              l = 1
c             Other centuries are not leap years
           else if (mod(iyear,100).eq.0) then
              l = 0
c             Otherwise, every fourth year is a leap year
           else if (mod(iyear,4).eq.0) then
              l = 1
c             Other years are not leap years
           else
              l = 0
           endif

c          Compute julian day (Annex 2 FAO-56)
           j = int(275.0*float(imon)/9.0-30.0+float(iday))-2

           if (float(imon).lt.3.0) then
                   j = j+2
           else if ((l.eq.1) .and. (float(imon).gt.2.0)) then
                   j = j+1
           endif

        else

           read (1, *, end=400) iyear,imon,iday,j,(dat(i),i=1,nproc)

        endif

c       Increment icount
        icount = icount+1

c       Set beginning year for data
        if (icount.eq.1) ibegyr=iyear

c       Check if this is a new year
        if (iyear.ne.iyrprev) then
          nydata=nydata+1
          years(nydata)=iyear
        endif

        iyrprev=iyear

c       Count data for each month
        ctm(imon)=ctm(imon)+1
        cty(nydata)=cty(nydata)+1
c        write(*,*) iyear,nydata,cty(nydata)

c       Loop through all stations accumulating data
        do 350 ista = 1,nproc
          s(imon,ista)=s(imon,ista)+dat(ista)
```

```
          sy(nydata,ista)=sy(nydata,ista)+dat(ista)
          sa(ista)=sa(ista)+dat(ista)
 350    continue

        go to 300
 400    continue

c       Loop through all months/stations computing means
        do imon=1,nm
          do ista=1,nproc
            if (ctm(imon).le.1) then
              write(*,*) 'Need at least 2 data points for month ', imon
              stop
            endif
            ave(imon,ista)=s(imon,ista)/ctm(imon)
          end do
        end do

c       Loop through all stations/years computing average annual
c       and year-to-year stdevs
        do ista=1,nproc
          avea(ista)=sa(ista)/nydata
          do iyear=1,nydata
            sdeva(ista)=sdeva(ista)
     a                  +(sy(iyear,ista)-avea(ista))**2
          end do
          if (nydata.le.1) then
            sdeva(ista)=-999.0
            cva(ista)=-999.0
          else
            sdeva(ista)=sqrt(sdeva(ista)/(nydata-1))
            cva(ista)=sdeva(ista)/avea(ista)
          endif
        end do

c       Read-in data again to compute other stats
        rewind 1
 301    continue
        if (fmt.eq.'1') then

           read (1, *, end=401) date,(dat(i),i=1,nproc)

c          Extract year, month, day from date
           iyear = date/10000
           imon = (date - iyear*10000)/100
           iday = (date - iyear*10000) - imon*100

c          Determine whether year is a leap year
c             Years divisible by 400 are leap years
           if (mod(iyear,400).eq.0) then
              l = 1
c             Other centuries are not leap years
           else if (mod(iyear,100).eq.0) then
              l = 0
c             Otherwise, every fourth year is a leap year
           else if (mod(iyear,4).eq.0) then
              l = 1
c             Other years are not leap years
           else
              l = 0
           endif

c          Compute julian day (Annex 2 FAO-56)
          j = int(275.0*float(imon)/9.0-30.0+float(iday))-2

          if (float(imon).lt.3.0) then
                  j = j+2
          else if ((l.eq.1) .and. (float(imon).gt.2.0)) then
                  j = j+1
          endif

        else
           read (1, *, end=401) iyear,imon,iday,j,(dat(i),i=1,nproc)
        endif
```

```
        write (*,*) 'Processing ',iyear,imon,iday,' (',j,' )'

c       Loop through all stations accumulating data
        do 351 ista = 1,nproc
          s(imon,ista)=dat(ista)-ave(imon,ista)
          adev(imon,ista)=adev(imon,ista)+abs(s(imon,ista))
          p(imon,ista)=s(imon,ista)**2
          var(imon,ista)=var(imon,ista)+p(imon,ista)
          p(imon,ista)=s(imon,ista)**3
         skew(imon,ista)=skew(imon,ista)+p(imon,ista)
         p(imon,ista)=s(imon,ista)**4
         kurt(imon,ista)=kurt(imon,ista)+p(imon,ista)
 351    continue

        go to 301
 401    continue

c       Loop through all months/stations computing stats
        do imon=1,nm
          do ista=1,nproc
            if (ctm(imon).le.1) then
              write(*,*) 'Need at least 2 data points for month ', imon
              stop
            endif
            adev(imon,ista)=adev(imon,ista)/ctm(imon)
            var(imon,ista)=var(imon,ista)/(ctm(imon)-1)
            sdev(imon,ista)=sqrt(var(imon,ista))
            if (var(imon,ista).ne.0.0) then
              skew(imon,ista)=
     a              skew(imon,ista)/(ctm(imon)*sdev(imon,ista)**3)
              kurt(imon,ista)=
     a              kurt(imon,ista)/(ctm(imon)*var(imon,ista)**2)-3.0
            else
c             No skew or kurtosis when zero variance
              skew(imon,ista)=-999.0
              kurt(imon,ista)=-999.0
            endif
            if (ave(imon,ista).ne.0.0) then
              cv(imon,ista)=sdev(imon,ista)/ave(imon,ista)
            else
              cv(imon,ista)=-999.0
           endif
          end do
        end do

c       Output stats for all stations, months
        do ista = 1,nproc
           write (11,50) (ave(imon,ista),imon=1,nm)
           write (12,50) (adev(imon,ista),imon=1,nm)
           write (13,50) (sdev(imon,ista),imon=1,nm)
           write (14,50) (var(imon,ista),imon=1,nm)
           write (15,50) (skew(imon,ista),imon=1,nm)
           write (16,50) (kurt(imon,ista),imon=1,nm)
           write (17,50) (cv(imon,ista),imon=1,nm)
 50        format (12(f10.5,2x))
        end do

c       Output annual data for all stations, years
        write (18,60) (years(iyear),iyear=1,nydata)
 60     format (200(4x,I4,2x,2x))
        do ista = 1,nproc
           if (aveflag.eq.'0') then
              write (18,70) (sy(iyear,ista),iyear=1,nydata)
           else
              write (18,70) (sy(iyear,ista)/cty(iyear),iyear=1,nydata)
           endif
 70        format (200(f10.5,2x))
        end do

c       Compute average number of days per year for period of record
        do iyear = 1,nydata
           avecty = avecty + cty(iyear)
        end do
        avecty = avecty / nydata
c       write (*,*) avecty, nydata
```

```
c       Output annual stats for all stations
        write (19,80) 'Ave','Sdev','Cv'
 80     format (3(4x,A4,2x,2x))
        do ista = 1,nproc
           if (aveflag.eq.'0') then
              write (19,90) avea(ista),sdeva(ista),cva(ista)
           else
              write (19,90) avea(ista)/avecty,sdeva(ista)/avecty,
     a                      cva(ista)
           endif
 90        format (3(f10.5,2x))
        end do

        close (1)
        close (11)
        close (12)
        close (13)
        close (14)
        close (15)
        close (16)
        close (17)
        close (18)
        close (19)

        stop
        end
```

## kernel2.f

```
         integer index(19)
         real xn(64),yn(64),xh(321),yh(321)
         real eth(18628,321),w(19),eth_N(18628,211)

c        open (unit=1,file='etopt_rhcap_hydro51.txt',status='old')
        open (unit=1,file='etopm_rhcap.txt',status='old')
        open(unit=2,file='hydro51_321_xy.csv',status='old')
        open(unit=3,file='NARR_xy_new.csv',status='old')
        open (unit=4,file='Agg_hydro_at_NARR_locs.out',status='unknown')

c  read NARR x, y
        read(3,*)
        read(3,*)
        do i=1,64
           read(3,*) xn(i),yn(i)
        enddo

c  read hydro51 x, y
        read(2,*)
        read(2,*)
        do i=1,321
           read(2,*) xh(i),yh(i)
        enddo
        close(2)
        nn=19
        do i=1,18628
                read(1,*)iy,im,id,idd,(eth(i,j),j=1,321)
        enddo
        close(1)

        do ip =1,64
                write(*,*)'now processing location ',ip

          x1=xn(ip)
          y1=yn(ip)

        open(unit=2,file='hydro51_321_xy.csv',status='old')
        read(2,*)
        read(2,*)
        do i=1,321
           read(2,*) xh(i),yh(i)
        enddo
        close(2)
          call xkernel(nn,x1,y1,xh,yh,b,index,w)
          sum=0.
          do i =1,nn
          sum=sum+w(i)
          enddo
 102    format(f8.5,1x,i4,1x,f8.5)
          do i=1,18628
            ee=0.
            do j=1,nn
                ee=ee+w(j)*eth(i,index(j))
            enddo
                eth_N(i,ip)=ee/sum
          enddo
        enddo
        open (unit=1,file='etopt_rhcap_hydro51.txt',status='old')
           write(4,201)(xn(ip),ip=1,64)
           write(4,201)(yn(ip),ip=1,64)
        do i=1,18628
        write(*,*)'now writing ',i
           read(1,*)iy,im,id,idd,(eth(i,j),j=1,321)
           write(4,200)iy,im,id,idd,(eth_N(i,ip),ip=1,64)
        enddo
201      format('x coordinates       ',64(f9.1,1x))
202      format('y coordinates       ',64(f9.1,1x))
200     format(i4,1x,i4,1x,i4,1x,i4,1x,64(f9.5,1x))
100     format(i4,1x,i3,1x,2(f10.2,1x),f8.5)
        stop
```

```
      end



      subroutine xkernel(nn,x1,y1,xh1,yh1,b,index,w)
       integer index(19)
      real xh1(321),yh1(321),dr(321),w(19)
       do i =1,321
              index(i)=i
              dx=x1-xh1(i)
              dy=y1-yh1(i)
              dr(i)=sqrt(dx*dx+dy*dy)
       enddo


      do i=1,nn
           do j=i+1,321
              if(dr(j).lt.dr(i)) then
                      temp = dr(i)
                      dr(i)=dr(j)
                      dr(j)=temp
                      itemp=index(i)
                      index(i)=index(j)
                      index(j)=itemp
              endif
           enddo
      enddo
      b=dr(nn)
      do i=1,nn
           dd=dr(i)/b
           w(i)=(1-dd*dd)*(1-dd*dd)
       write(18,101)b,index(i),w(i),xh1(index(i)),yh1(index(i))
      enddo
101   format(f10.2,1x,i4,1x,f8.5,1x,2(f10.2,1x))
      return
      end
```

## rescale99.f

```
c     Program: rescale.f
c          By: Tim Newton
c        Date: 25-Jul-2006
c Description: This program rescales Hydro51 forcing PET 1948-98
c
c     Compile: f77 -o rescale.exe rescale.f
c
c=======================================================================
c Dimension variables
c=======================================================================
c HP=H', pet=etopm_rhcap value, HA=average H,sdh=std dev H, sdn=std dev NARR
c NA= Average NARR
      real HP(325),HA(325,13),sdh(325,13),NA(325,13),sdn(325,13)
      real pet(325)
      integer ct, j, k, n
      integer mo,da,yr,jda
c=======================================================================
c Open files to read and write
c=======================================================================

c-----Copied from
\\oomdata\ws\NSRSM\data\PET\PYTHON_OUT\NARR\Final_PM_ETo\Stats\PenMon_SWR_75\LandPts\ave_
clim99.txt
      open(10,file='ave_clim99.txt',status='old')
        do 15 ct = 1, 99
c-----------Read variables into array
             read(10,*) (NA(ct,n),n=1,12)
  15         continue

c-----Copied from
\\oomdata\ws\NSRSM\data\PET\PYTHON_OUT\NARR\Final_PM_ETo\Stats\PenMon_SWR_75\LandPts\sdev
_clim99.txt
      open(20,file='sdev_clim99.txt',status='old')
        do 20 j = 1, 99
c-----------Read variables into array
              read(20,*) (sdn(j,n),n=1,12)
  20         continue

      open(30,file='./Stats/ave_clim.txt',status='old')
        do 30 ct = 1, 99
c-----------Read variables into array
              read(30,*) (HA(ct,n),n=1,12)
  30         continue

      open(40,file='./Stats/sdev_clim.txt',status='old')
        do 40 j = 1, 99
c-----------Read variables into array
              read(40,*) (sdh(j,n),n=1,12)
  40         continue

      open(50,file='HYDRO51_PRIME.dat',status='unknown')

      open(60,file='./Stats/data.txt',status='old')

c=======================================================================
c Begin BIG loop
c From 1948 to 1998, 51 years
c=======================================================================

        do 50 ct = 1, 18628

c-----------Read variables into array
              read(60,*) yr,mo,da,jda,(pet(n),n=1,99)
              do 60 k = 1, 99
      HP(k)=((pet(k)-HA(k,mo))/sdh(k,mo))*sdn(k,mo)+NA(k,mo)

 60         continue

            write(50,1025) yr,mo,da,jda,(HP(n),n=1,99)
```

```
 50         continue

            close(50)


c=======================================================================
c Format statements
c=======================================================================
 1025 format (i4," ",i2," ",i2," ",i3," ",99(f6.4,2x))

c=======================================================================
c End program
c=======================================================================
      stop
      end
```

## merge_datasets2.scr

```
#!/bin/csh -f
#Script to merge Hydro51' and NARR datasets and create a DSS file for use in gr_thsn
program
#Michelle M. Irizarry-Ortiz 08/21/06

set dssfilename = Hydro51P_NARR_PMETo_RHcap.dss

set HPbegyr = 1948
set NARRbegyr = 1979
set NARRendyr = 2005

echo $dssfilename >! temp1.$$

gawk '{if(NR>1){print $2}}' /nw/oomdata_ws/NSRSM/data/PET/Rescaling/NARR_xy_99.prn >
NARR_active_ids

#Create file with dates
gawk '{if($1<cutyear)print $1,$2,$3}' cutyear=$NARRbegyr
/nw/oomdata_ws/NSRSM/data/PET/Rescaling/HYDRO51_PRIME.dat > dates.txt

gawk '{print $1,$2,$3}'
/nw/oomdata_ws/NSRSM/data/PET/PYTHON_OUT/NARR/Final_PM_ETo/Stats/PenMon_SWR_75/data.txt
>> dates.txt

#Go through each station merging Hydro51' and NARR data
@ i = 0
foreach id (`cat NARR_active_ids`)
  @ i++
  echo id $id $i

  set colH = `expr $i + 4`
  gawk '{if($1<cutyear)print $col}' cutyear=$NARRbegyr col=$colH
/nw/oomdata_ws/NSRSM/data/PET/Rescaling/HYDRO51_PRIME.dat > temp1.txt

  set colN = `expr $id + 4`
  gawk '{print $col}' col=$colN
/nw/oomdata_ws/NSRSM/data/PET/PYTHON_OUT/NARR/Final_PM_ETo/Stats/PenMon_SWR_75/data.txt
>> temp1.txt

  paste dates.txt temp1.txt > temp2.txt

  #Create header part of text file for storage into DSS
  set row = `expr $i + 1`
  gawk 'BEGIN{print "TIME WINDOW 01JAN1948 31DEC2005\n"};{if(NR>1){printf "STATION
\"%s\"\nXCOORD %s\nYCOORD %s\nDBKEY \"CALC-%s\"\nBASIN \"outdomain\"\nLOCATION
\"%s\"\nPARAMETER \"PM_ETo\"\nINTERVAL \"DA\"\nDESCRIPTOR \"%s\"\nALT_ID \"\"\nAGENCY
\"SFWMD\"\nUNITS
\"INCHES\"\nDSSFILE\"Hydro51P_NARR_PMETo_RHcap\"\nQUALFLAGS\nEND\n\n",$2,$5,$6,$2,$2,$2}}
;END{print "DATA"}' /nw/oomdata_ws/NSRSM/data/PET/Rescaling/NARR_xy_99.prn > header.txt

  cat header.txt temp2.txt > temp3.txt
  dos2unix temp3.txt > fordss.${i}.txt

  ./Sto_tnewton fordss.${i}.txt

end

/bin/rm temp*.txt temp*.$$ header.txt
```