Handling of missing data in RPN standard files
Yves Chartier – February 2005

It happens sometimes that a user wants to define some parts of gridded data as missing. The following documents provides some suggestions about how to handle this task in RPN standard files.

It is a fairly common practive in FORTRAN programs to use a special numerical value to flag missing data. For instance, one might use a numerical code (eg. 999.0) to define areas of the grid where a given field (say, temperature) data is missing. Unless such a data record is encoded without compression (ie X32 or E32), this is generally a bad idea, for two reasons :

- With the compression schema currently used in the standard files, the numerical value used to flag the missing data has good chances to be different than the one originally encoded, especially if this value is not the minimum value of the field.
- The precision of the whole dataset will be severely affected since it will artificially expand its numerical range, meaning that fewer bits will be available to encode the valid portion of the data.

An implicit method that has been suggested to encode missing values - and that recent versions of XREC were trained to recognize - is to define a special value according to the following formula

$$SPVAL = max + 0.1 * (max - min)$$

For instance, if a temperature field has a maximum value of 100.0, a minimum value of 0.0, then SPVAL = 110.0.

However a better scheme is desired, if only for the case where some data fields may exist where the difference between the maximum and 2nd maximum values exceed 10 % of the range of the field.

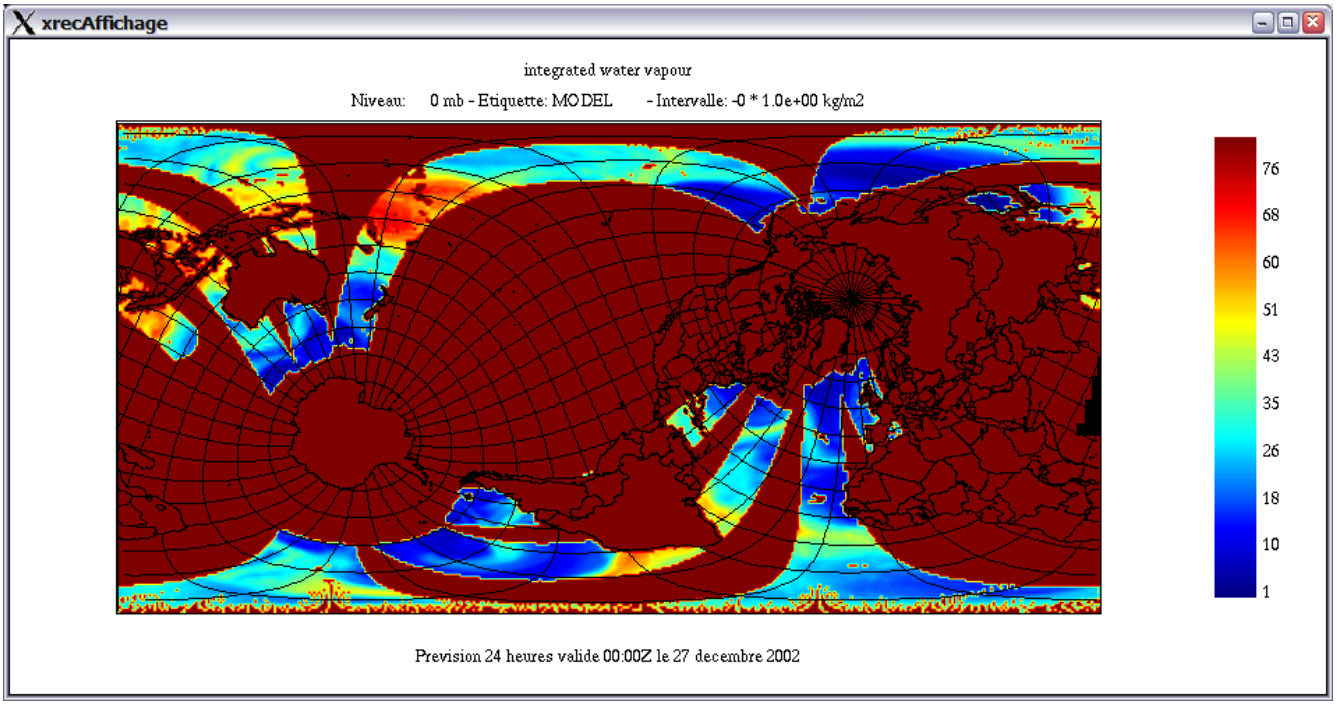
The now recommanded procedure to encode missing values is as follows :

1. Fields with missing values have a special TYPVAR variable, in which the 2nd character is @ (eg. TYPVAR=P becomes TYPVAR=P@)
2. These fields have a companion field, a bitmask indicating the absence or presence of data. The bitmask has exactly all the attributes of the master field, with the following exceptions :
   - TYPVAR = @@
   - datyp = 2 (unsigned integer)
   - nbits = 1
   - In this field, absence of data (missing) = 0, and presence of data (non-missing) = 1

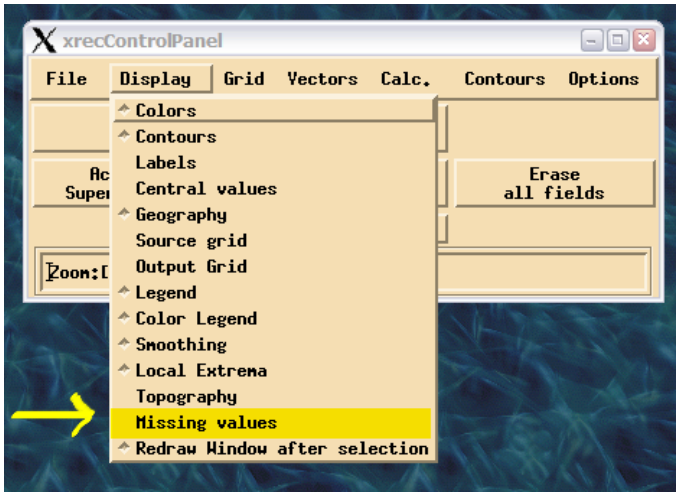
Here is an exemple from xvoir on a sample dataset.

```
IH  P@    0 mb      24    0   401   200     1 MODEL       20021226 000000    2700    32    Z    1001.0    1023.0      0.0      0.0  R12
IH  @@    0 mb      24    0   401   200     1 MODEL       20021226 000000    2700    32    Z    1001.0    1023.0      0.0      0.0  I01
```

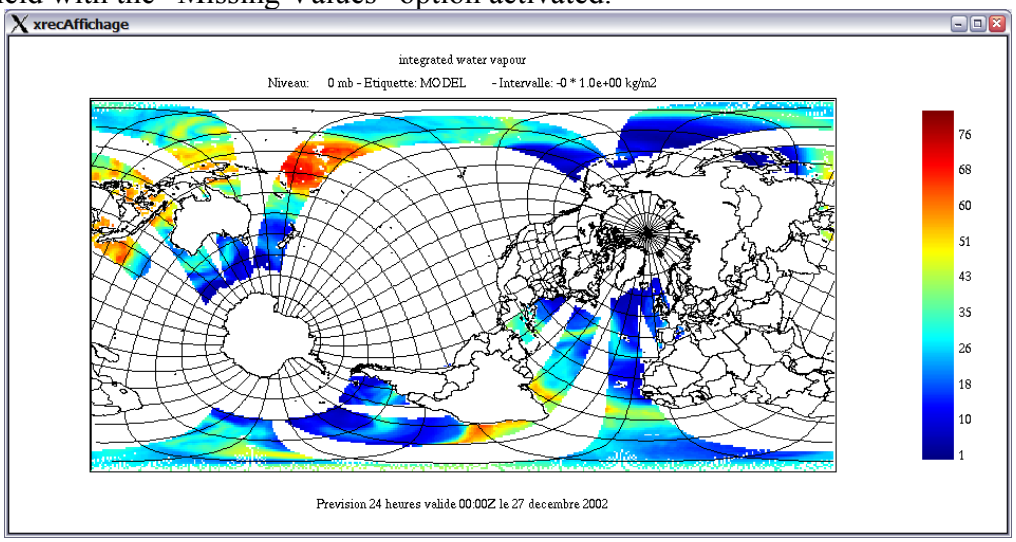
This is how a field with missing values looks in XREC when the “Missing Values” option is not activated from the “Display” menu.



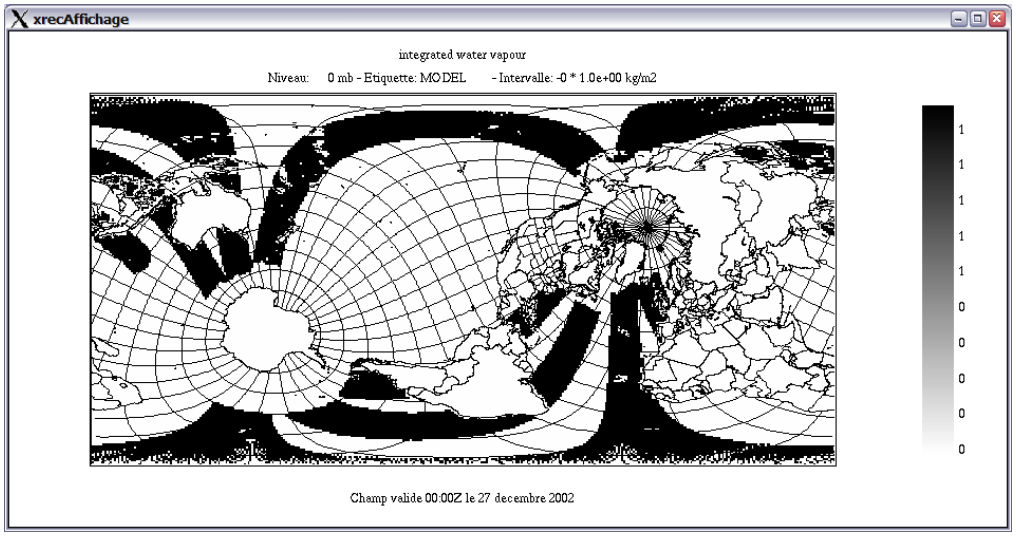
This shows how to activate the “Missing Values” option from xrec5.3



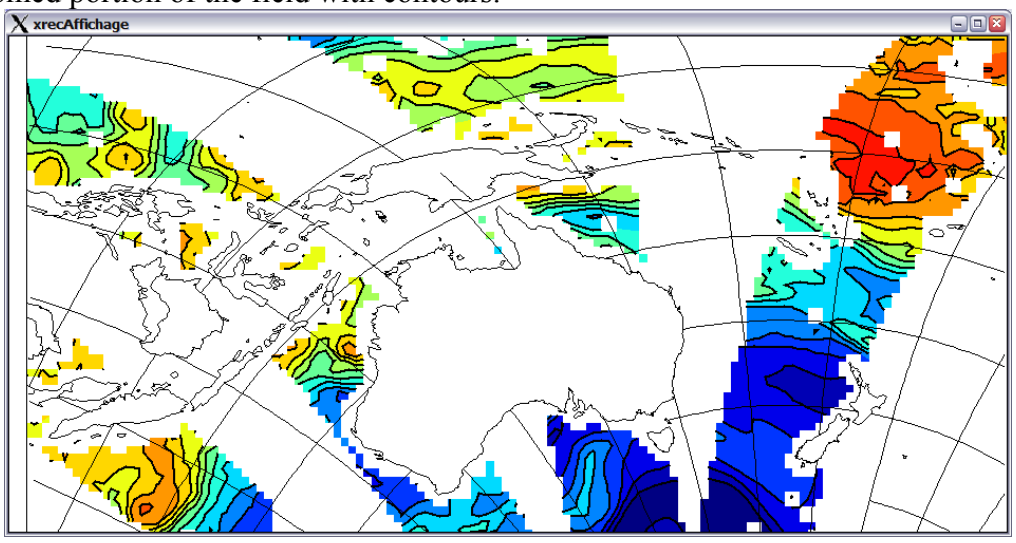
This is the field with the “Missing Values” option activated.



This is the bitmask.

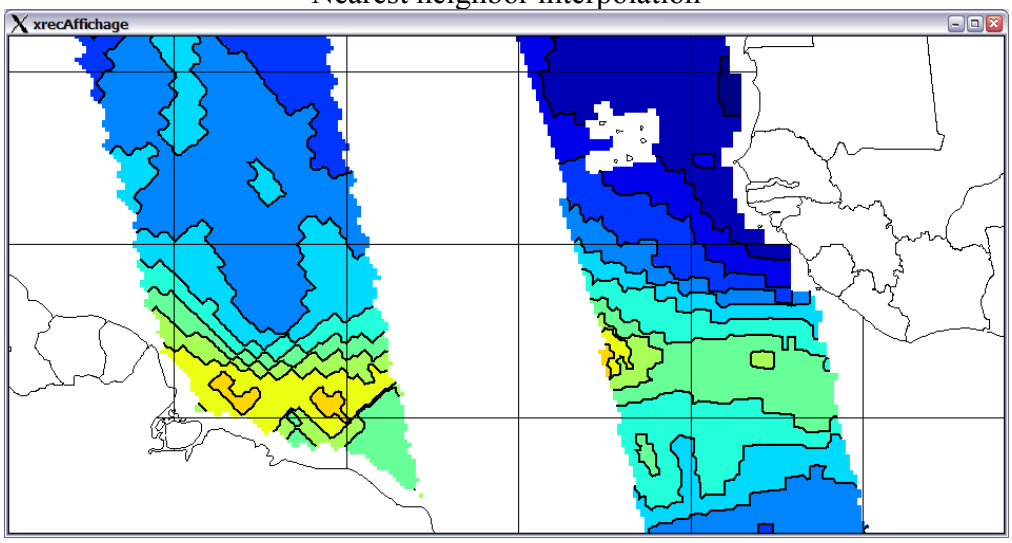


This is a zoomed portion of the field with contours.

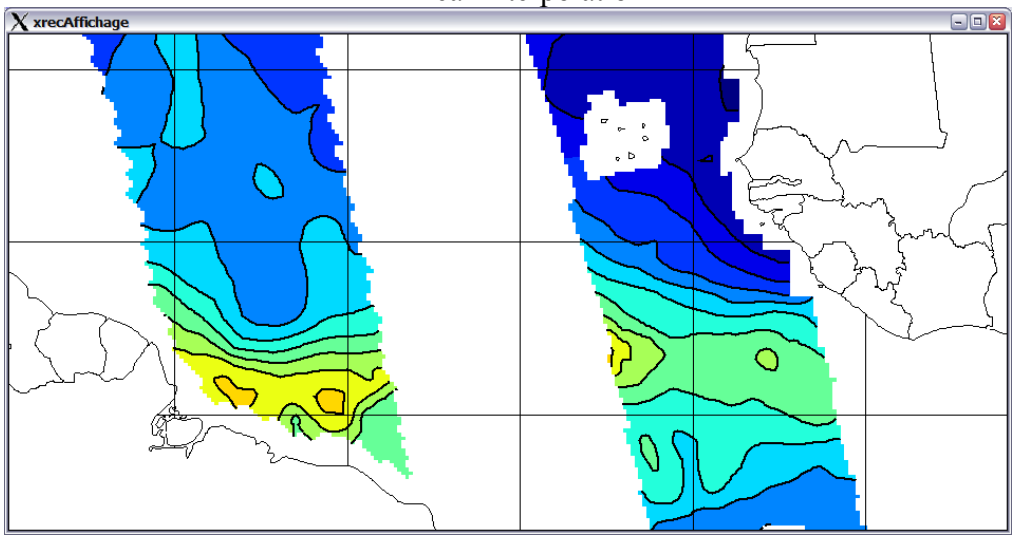


When interpolating missing values onto a target grid, it needs only one cell with missing data among those from the source field to invalidate the interpolation result. This restriction is the most severe with bicubic interpolation, where 16 grid points from the source grid are needed.The following pictures show the effect of the mask when interpolating fields with missing values onto another grid.

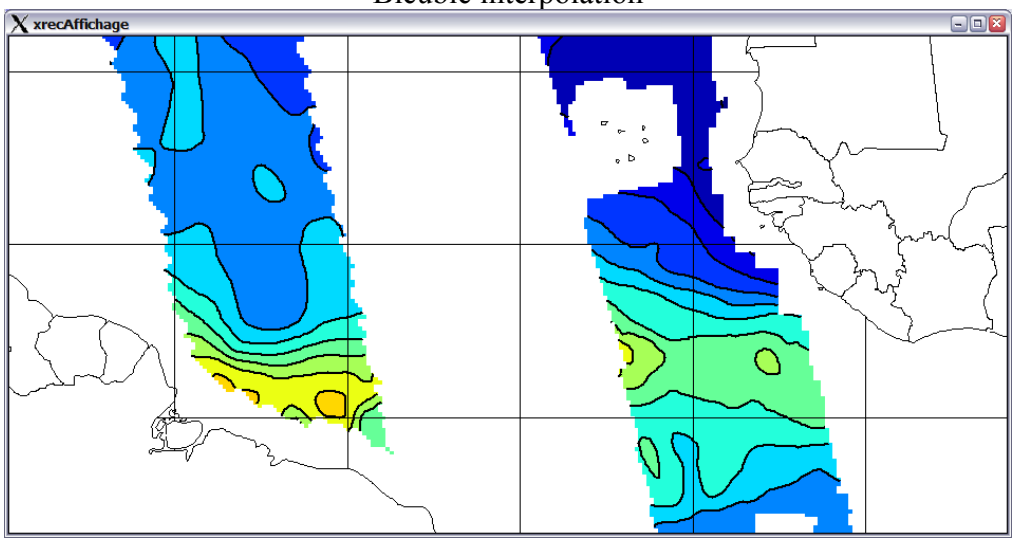
Nearest neighbor interpolation



Bilinear interpolation



Bicubic interpolation



How to encode the masks in FORTRAN

The following code is an excerpt from a program that writes a mask from a field where the missing values have been encoded with the method

$$SPVAL = max + 0.1 * (max - min)$$

```
! Open the input (unit 1) and output (unit 2) files

  iunin = 1
  iunout = 2
  ier = fnom(iunin,  val(1),'RND+OLD+R/O',0)
  ier = fnom(iunout, val(2),'RND',0)
  ier = fstouv(iunin, 'RND')
  ier = fstouv(iunout, 'RND')

! Initialize the FST parameters attributes to collect all the fields

  datev = -1
  ip1  = -1
  ip2  = -1
  ip3  = -1
  etiket = '            '
  typvar = '  '
  nomvar = '    '

! Loop on all the fields found

  key = fstinf(iunin, ni, nj, nk,  datev, etiket, ip1,ip2,ip3,typvar,nomvar)
  do while (key.ge.0)

    ier = fstprm(key, dateo, deet, npas, ni, nj, nk, nbits, &
        datyp, ip1, ip2, ip3, typvar, nomvar, etiket2, grtyp, &
        ig1, ig2, ig3, ig4, swa, lng, dltf, ubc, &
        extra1, extra2, extra3)
    allocate(buf(ni,nj))
    allocate(masque(ni,nj))
    ier = fstluk(buf,key,ni,nj,nk)

! The “sminmax2” function returns the 1st and 2nd min and max values found in the field

    call sminmax2(rmin, rmax, rmin1, rmax1, buf, ni, nj, 1, 1, ni, nj)
    print *, rmin, rmin1, rmax1, rmax

    threshold = rmax1 + 0.1 * (rmax1 - rmin)

! Look if the “missing value” criterion is found and fills the mask values accordingly

    if (threshold < 1.001*rmax .and. threshold > 0.999 * rmax) then
      print *, 'Trouve...', threshold, rmax, rmax-threshold,rmax*1.001, rmax*0.999
      typvarvm(1:1) = typvar(1:1)
      typvarvm(2:2) = '@'
      typvarm     = '@@'
      do j=1,nj
        do i=1,ni
          if (buf(i,j) .eq. rmax) then
            masque(i,j) = 0
          else
            masque(i,j) = 1
          endif
        enddo
      enddo
      ier = fstecr(buf,unused,nbits,iunout,dateo,deet,npas,ni,nj,nk,ip1,ip2,ip3,
                   typvarvm,nomvar,etiket2,grtyp,ig1,ig2,ig3,ig4,1,.false.)
      ier = fstecr(masque,unused,1,iunout,dateo,deet,npas,ni,nj,nk,ip1,ip2,ip3,
                   typvarm,nomvar,etiket2,grtyp,ig1,ig2,ig3,ig4,2,.false.)
    else

! If the treshold is not met then just rewrite the field as is

      ier = fstecr(buf,unused,-nbits,iunout,dateo,deet,npas,ni,nj,nk,ip1,ip2,ip3,
                   typvar,nomvar,etiket2,grtyp,ig1,ig2,ig3,ig4,1,.false.)
    endif

    key = fstsui(iunin,ni,nj,nk)
    deallocate(buf)
    deallocate(masque)
  end do
```