Contents

1 Introduction

GRID is a FORTRAN subroutine library that contains some routines to read and write DEOS-formatted grids.

The GRID library has been ported to various systems. Presently it is available at DEOS/LR on the IBM Risk 6000 and the SGI servers. At NOAA it is available for Linux.

The library can be linked to any of your FORTRAN programs by specifying the library in you FORTRAN link step, *e.g.*

```
f77 example.f -o example $ALTIM/lib/grid.a
```

The GRID library is courtesy of Remko Scharroo, Delft University of Technology, Delft Institute for Earth-Oriented Space Research. (E-mail: `remko.scharroo@lr.tudelft.nl`)

2 Subroutine Synopsis

2.1 DYNTOPO – Compute velocity from dynamic topography

```
    SUBROUTINE DYNTOPO (Z, VX, VY, NX, NY, MX, X0, X1, Y0, Y1)
    INTEGER NX, NY, MX
    REAL Z(MX,*), VX(MX,*), VY(MX,*), X0, X1, Y0, Y1
```

```
Determine velocity field components VX (due West) and VY (due North)
based on the dynamic topography height Z. The velocity grids and
the dynamic topography grid have the same resolution and are for the
same area.

Arguments:
 Z       (input): grid of dynamic height (unit: metres)
 VX, VY (output): grids of X and Y component of the velocity field
                  (unit: metres per second)
 NX, NY  (input): dimension of grids
 MX      (input): first dimension of arrays Z, VX, and VY
 X0, X1  (input): longitude boundaries (deg)
 Y0, Y1  (input): latitude boundaries (deg)
```

2.2 GR1INT4 – Interpolate linearly in a two-dimensional array

```
    FUNCTION  GR1INT4 (Z, X, Y, NX, NY, MX)
    INTEGER*4 NX, NY, MX
    REAL*4    GR1INT4, Z(MX,*), X, Y
```

```
This subroutine interpolates linearly in a two-dimensional regular grid.
The dimension of the grid is NX by NY. The real coordinates of the point to be
interpolated are (X,Y) where 1 <= X <= NX and 1 <= Y <= NY.

Arguments:
Z       (input): Two dimension array containing grid.
X,Y     (input): REAL*4 grid coordinates of the point to be interpolated.
NX,NY   (input): X- and Y- dimension of the grid.
MX      (input): First dimension of the array as defined in the calling
                 routine. MX >= NX.
GR1INT4 (output): Linearly interpolated value at point (X,Y).
```

2.3 GR1INT – Interpolate linearly in a two-dimensional array

```
    FUNCTION  GR1INT (Z, X, Y, NX, NY, MX)
    INTEGER*4 NX, NY, MX
    REAL*8    GR1INT, Z(MX,*), X, Y
```

This subroutine interpolates linearly in a two-dimensional regular grid.
The dimension of the grid is NX by NY. The real coordinates of the point to be interpolated are (X,Y) where 1 <= X <= NX and 1 <= Y <= NY.

Arguments:

Z	(input):	Two dimension array containing grid.
X,Y	(input):	REAL*8 grid coordinates of the point to be interpolated.
NX,NY	(input):	X- and Y- dimension of the grid.
MX	(input):	First dimension of the array as defined in the calling routine. MX >= NX.
GR1INT	(output):	Linearly interpolated value at point (X,Y).

2.4 GR2INT4 – Interpolate quadratically in a two-dimensional array

```
    FUNCTION  GR2INT4 (GRID, X, Y, NX, NY, MX)
    INTEGER*4 NX, NY, MX
    REAL*4    GR2INT4, GRID(MX,*), X, Y
```

This subroutine interpolates quadratically in a two-dimensional regular grid.
The dimension of the grid is NX by NY. The real coordinates of the point to be interpolated are (X,Y) where 1 <= X <= NX and 1 <= Y <= NY.

Arguments:

GRID	(input):	Two dimension array containing grid.
X,Y	(input):	REAL*4 grid coordinates of the point to be interpolated.
NX,NY	(input):	X- and Y- dimension of the grid.
MX	(input):	First dimension of the array as defined in the calling routine. MX >= NX.
GR2INT4	(output):	Quadratically interpolated value at point (X,Y).

2.5 GR2INT – Interpolate quadratically in a two-dimensional array

```
    FUNCTION  GR2INT (GRID, X, Y, NX, NY, MX)
    INTEGER*4 NX, NY, MX
    REAL*8    GR2INT, GRID(MX,*), X, Y
```

```
This subroutine interpolates quadratically in a two-dimensional regular grid.
The dimension of the grid is NX by NY. The real coordinates of the point to be
interpolated are (X,Y) where 1 <= X <= NX and 1 <= Y <= NY.

Arguments:
GRID    (input): Two dimension array containing grid.
X,Y     (input): REAL*8 grid coordinates of the point to be interpolated.
NX,NY   (input): X- and Y- dimension of the grid.
MX      (input): First dimension of the array as defined in the calling
                 routine. MX >= NX.
GR2INT (output): Quadratically interpolated value at point (X,Y).
```

2.6 GR3INT4 – Interpolate cubically in a two-dimensional REAL*4 array

```
    FUNCTION  GR3INT4 (GRID, X, Y, NX, NY, MX)
    INTEGER*4 NX, NY, MX
    REAL*4    GR3INT4, GRID(MX,*), X, Y
```

```
This subroutine interpolates cubically in a two-dimensional regular grid.
The dimension of the grid is NX by NY. The real coordinates of the point to be
interpolated are (X,Y) where 1 <= X <= NX and 1 <= Y <= NY.

Arguments:
GRID    (input): Two dimension array containing grid.
X,Y     (input): REAL*4 grid coordinates of the point to be interpolated.
NX,NY   (input): X- and Y- dimension of the grid.
MX      (input): First dimension of the array as defined in the calling
                 routine. MX >= NX.
GR3INT4 (output): Cubically interpolated value at point (X,Y).
```

2.7 GR3INT – Interpolate cubically in a two-dimensional REAL*8 array

```
    FUNCTION  GR3INT (GRID, X, Y, NX, NY, MX)
    INTEGER*4 NX, NY, MX
    REAL*8    GR3INT, GRID(MX,*), X, Y
```

```
This subroutine interpolates cubically in a two-dimensional regular grid.
The dimension of the grid is NX by NY. The real coordinates of the point to be
interpolated are (X,Y) where 1 <= X <= NX and 1 <= Y <= NY.

Arguments:
GRID    (input): Two dimension array containing grid.
X,Y     (input): REAL*8 grid coordinates of the point to be interpolated.
NX,NY   (input): X- and Y- dimension of the grid.
MX      (input): First dimension of the array as defined in the calling
                 routine. MX >= NX.
GR3INT (output): Cubically interpolated value at point (X,Y).
```

2.8 GRIDBINF – Return information about buffered grid

```
    SUBROUTINE GRIDBINF (POINTER, NX, NY, XMIN, XMAX, YMIN, YMAX,
   |                     ZMIN, ZMAX)
    INTEGER*4 POINTER, NX, NY
    REAL*8    XMIN, XMAX, YMIN, YMAX, ZMIN, ZMAX
```

```
This routine returns information about a grid previously loaded with
the subroutine GRIDBUFF.

Input argument:
 POINTER    : Pointer to grid structure as returned by GRIDBUFF

Output arguments:
 NX, NY     : Number of gridpoints in X- and Y-direction
 XMIN, XMAX : X-coordinate of the left- and rightmost gridpoint
 YMIN, YMAX : Y-coordinate of the lower- and uppermost gridpoint
 ZMIN, ZMAX : Minimum and maximum value in the grid
```

2.9 GRIDBINQ – Look-up value in buffered grid

```
FUNCTION GRIDBINQ (POINTER, X, Y)
INTEGER*4 POINTER
REAL*8 GRIDBINQ, X, Y
```

```
This function looks up a single value in a buffered grid that
was previously loaded using GRIDBUFF. No interpolation is
performed, the value at the closest grid point is returned.

The location at which the grid is to be queries is given by X and
Y. These arguments are given in "world coordinates"; in other words, X
must be between XMIN and XMAX, the coordinates of the left- and
right-most grid point. Something similar holds for the Y-coordinate.

Upon exit, the function value GRIDBINQ will be the value
of the grid at a grid node closest to (X, Y). When that point is
undetermined or when X and/or Y are out of limits, GRIDBINQ
will return a NaN value.

Input arguments:
 POINTER  : Pointer to the grid structure as returned by GRIDBUFF
 X, Y     : X- and Y-coordinate of the point to be interpolated

Output argument:
 GRIDBINQ : Value at the location (X, Y)
```

2.10 GRIDBINT – Bi-linear interpolation of buffered grid

```
FUNCTION GRIDBINT (POINTER, X, Y)
INTEGER*4 POINTER
REAL*8 GRIDBINT, X, Y
```

This function interpolates a buffered grid that was previously loaded using GRIDBUFF. Bi-linear interpolation is used whenever possible.

The location at which the grid is to be interpolated is given by X and Y. These arguments are given in "world coordinates"; in other words, X must be between XMIN and XMAX, the coordinates of the left- and right-most grid point. Something similar holds for the Y-coordinate.

Upon exit, the function value GRIDBINT will be the interpolated value of the grid at the location (X, Y). When (X, Y) points directly to a grid point, the value at this grid point is returned, otherwise bi-linear interpolation is performed between the four grid points surrounding (X, Y). When one of the grid points is not determined, a triangular interpolation is conducted.

When X and/or Y are out of the limits of the grid, or when (X, Y) is close to an undetermined grid point, GRIDBINT will return a NaN value.

Input arguments:
 POINTER : Pointer to the grid structure as returned by GRIDBUFF
 X, Y   : X- and Y-coordinate of the point to be interpolated

Output argument:
 GRIDBINT : Interpolated value at the location (X, Y)

2.11 GRIDBSPL – Bi-cubic spline interpolation of buffered grid

```
FUNCTION GRIDBSPL (POINTER, X, Y)
INTEGER*4 POINTER
REAL*8 GRIDBSPL, X, Y
```

```
This function interpolates a buffered grid that was previously loaded
using GRIDBUFF. Bi-cubic spline interpolation is used whenever possible.

The location at which the grid is to be interpolated is given by X and
Y. These arguments are given in "world coordinates"; in other words, X
must be between XMIN and XMAX, the coordinates of the left- and
right-most grid point. Something similar holds for the Y-coordinate.

Upon exit, the function value GRIDBSPL will be the interpolated value
of the grid at the location (X, Y). When (X, Y) points directly to a
grid point, the value at this grid point is returned, otherwise
bi-cubic spline interpolation is performed between the 6-by-6 grid
points surrounding (X, Y).

This routine DOES NOT take into account invalid values. Hence, it
assumes that all values in the 6-by-6 subgrid are valid. This will
work for most geoid and mean sea surface grids.

When X and/or Y are out of the limits of the grid, or too close to the
boundary in order to perform bi-cubic spline interpolation,
GRIDBSPL will return a NaN value.

Input arguments:
 POINTER : Pointer to the grid structure as returned by GRIDBUFF
 X, Y    : X- and Y-coordinate of the point to be interpolated

Output argument:
 GRIDBSPL : Interpolated value at the location (X, Y)
```

2.12 GRIDBUFF – Load a grid into memory

```
    FUNCTION GRIDBUFF (FILENM, POINTER)
    INTEGER*4 GRIDBUFF, POINTER
    CHARACTER FILENM*(*)
```

```
This routine allocates memory and loads the contents of a grid file
into the allocated memory. In contrast to the GRIDRD4 and GRIDRD8
routines, the grid values are not directly available.

Other information on the grid, like grid dimension, can be obtained
with the GRIDBINF routine.

After using GRIDBUFF, GRIDBINT or GRIDBSPL can be used to interpolate
the grid, based on bi-linear or bi-cubic spline interpolation.

The GRIDBUFF routine handles grids of either 2-, 3- or 4-byte integers,
in little or big endian notation, on little or big endian machines.
These grids can be recognised by their @GRID, @GR2L, @GR2B, @GR3L,
@GR3B, @GR4L or @GR4B preamble.

GRIDBUFF is also more efficient in the memory use than GRIDRD4 and
GRIDRD8. In case of 2-byte grids, it uses almost half the memory
GRIDRD4 uses, and about a quarter of the memory GRIDRD8 needs.
This is achieved by loading the original 2-byte-per-cell grid into
memory without further manipulation. The value returned by GRIDBUFF
is a pointer to a structure that includes both the grid dimensions
and the grid itself.

When the allocation of memory or the loading of the grid was
unsuccessful, this will be reflected in the returned pointer value.

Input argument:
 FILENM   : Name of the file containing the grid.

Output argument:
 POINTER  : Pointer to the grid structure

Return value:
 GRIDBUFF : 0 = No error
            1 = Could not find or open grid
            2 = Ilegal grid format
            3 = Memory allocation was unsuccessful
            4 = Error loading grid
            5 = Wrong specifier
```

2.13 GRIDCOG4 – Find the Centre-Of-Gravity in a REAL*4 grid

```
SUBROUTINE GRIDCOG4 (Z, NX, NY, MX, ZMIN, ZMAX, XCOG, YCOG)
INTEGER*4 NX, NY, MX
REAL*4    Z(MX,NY), ZMIN, ZMAX, XCOG(NY+1), YCOG(NX+1)
```

This routine termines the Centre-Of-Gravity (COG) of a two dimensional matrix (grid) of REAL*4 values. Along with it, it determined the COG along each gridline in both directions.

The COG along a vertical gridline I is defined as SUM Z(I,J)*J / SUM Z(I,J), where J runs from 1 to NY. A similar definition holds for the COG along a horizontal gridline. Values of Z beyond the limits ZMIN and ZMAX are not taken into account.

The position of the COG in vertical direction is stored in YCOG, horizontal in XCOG.

The total COG of the matrix is found by taking into account all gridlines. The X and Y coordinates are stored as XCOG(NY+1) and YCOG(NX+1).

Coordinates of the COG are expressed in gridpoints, counting from 1 to NX or 1 to NY for horizontal and vertical coordinates respectively. A value of 0 is stored when the COG could not be computed (division by zero).

Arguments:

Z	(input):	Input matrix (grid)
NX	(input):	Number of gridpoints in X (horizontal) direction
NY	(input):	Number of gridpoints in Y (vertical) direction
MX	(input):	First dimension of Z as determined in the calling program or routine
ZMIN	(input):	Lower limit of Z for editing
ZMAX	(input):	Upper limit of Z for editing
XCOG	(output):	Horizontal positions of the COG. XCOG(J) contains the value for vertical gridline J, XCOG(NY+1) for the entire matrix
YCOG	(output):	Vertical positions of the COG. YCOG(I) contains the value for horizontal gridline I, YCOG(NX+1) for the entire matrix

2.14 GRIDCOG8 – Find the Centre-Of-Gravity in a REAL*8 grid

```
SUBROUTINE GRIDCOG8 (Z, NX, NY, MX, ZMIN, ZMAX, XCOG, YCOG)
INTEGER*4 NX, NY, MX
REAL*8    Z(MX,NY), ZMIN, ZMAX, XCOG(NY+1), YCOG(NX+1)
```

This routine termines the Centre-Of-Gravity (COG) of a two dimensional matrix (grid) of REAL*8 values. Along with it, it determined the COG along each gridline in both directions.

The COG along a vertical gridline I is defined as SUM Z(I,J)*J / SUM Z(I,J), where J runs from 1 to NY. A similar definition holds for the COG along a horizontal gridline. Values of Z beyond the limits ZMIN and ZMAX are not taken into account.

The position of the COG in vertical direction is stored in YCOG, horizontal in XCOG.

The total COG of the matrix is found by taking into account all gridlines. The X and Y coordinates are stored as XCOG(NY+1) and YCOG(NX+1).

Coordinates of the COG are expressed in gridpoints, counting from 1 to NX or 1 to NY for horizontal and vertical coordinates respectively. A value of 0 is stored when the COG could not be computed (division by zero).

Arguments:

Z	(input):	Input matrix (grid)
NX	(input):	Number of gridpoints in X (horizontal) direction
NY	(input):	Number of gridpoints in Y (vertical) direction
MX	(input):	First dimension of Z as determined in the calling program or routine
ZMIN	(input):	Lower limit of Z for editing
ZMAX	(input):	Upper limit of Z for editing
XCOG	(output):	Horizontal positions of the COG. XCOG(J) contains the value for vertical gridline J, XCOG(NY+1) for the entire matrix
YCOG	(output):	Vertical positions of the COG. YCOG(I) contains the value for horizontal gridline I, YCOG(NX+1) for the entire matrix

2.15 GRIDDX – Derivative of grid in X derection

```
    SUBROUTINE GRIDDX (H, G, NX, NY, MX)
    INTEGER NX, NY, MX
    REAL H(MX,*), G(MX,*)

Compute derivative of geometry grid H along the X axis

Arguments:
 H     (input): Geometry grid
 G    (output): Derivative grid (DH/DX)
 NX,NY (input): Dimension of the grid
 MX    (input): First dimension of arrays H and G
```

2.16 GRIDDY – Derivative of grid in X derection

```
    SUBROUTINE GRIDDY (H, G, NX, NY, MX)
    INTEGER NX, NY, MX
    REAL H(MX,*), G(MX,*)

Compute derivative of geometry grid H along the X axis

Arguments:
 H     (input): Geometry grid
 G    (output): Derivative grid (DH/DX)
 NX,NY (input): Dimension of the grid
 MX    (input): First dimension of arrays H and G
```

2.17 GRIDRD4 – Read grid into a REAL*4 array

```
    FUNCTION GRIDRD4 (FILENM, NX, NY, Z,
   .                  XMIN, XMAX, YMIN, YMAX, ZMIN, ZMAX)
    CHARACTER FILENM*(*)
    INTEGER*4 NX, NY, GRIDRD4
    REAL*4    Z(*), XMIN, XMAX, YMIN, YMAX, ZMIN, ZMAX
```

This routine reads the contents of a grid file into a REAL*4 array.
This array specified in the calling routine may be one- or two-dimensional.
The values will be stored row by row, starting at the lower left corner,
and pertain to the nodes of the grid. The coordinates of the value
stored in Z(KX,KY) will thus be:

```
  X = XMIN + (KX-1) * DX   with   DX = (XMAX-XMIN) / (NX-1)
  Y = YMIN + (KY-1) * DY   with   DY = (YMAX-YMIN) / (NY-1)
```

were NX and NY are the actual integer dimensions of the grid. The area
covered by the grid is always rectangular and has corners (XMIN,YMIN) and
(XMAX,YMAX).
If the array is specified one-dimensional in the calling routine, you may
specify an original X-dimension of 0 (NX=0) and NY equal to the maximum
size of the single dimension. In that case, the array will be filled as
were the original dimensions of the array equal to the actual dimensions
of the grid.
Points that do not have a defined value in the grid are given a value of
1.0D+35.

Arguments:

FILENM	(input):	Name of the file containing the grid.
NX	(input):	X-dimension of array Z as defined in the calling routine. If NX=0, Z is defined one-dimensional.
	(output):	Actual X-dimension of the grid.
NY	(input):	Y-dimension of array Z as defined in the calling routine. If NX=0, NY specifies the maximum single dimension.
	(output):	Actual Y-dimension of the grid, or zero if error occurred.
Z	(output):	Array into which the gridded data must be stored.
XMIN	(output):	X (longitude) value of lower left corner.
XMAX	(output):	X (longitude) value of upper right corner.
YMIN	(output):	Y (latitude) value of lower left corner.
YMAX	(output):	Y (latitude) value of upper right corner.
ZMIN	(output):	minimum Z (height) value in the grid.
ZMAX	(output):	maximum Z (height) value in the grid.
GRIDRD4	(output):	Return code: =0 : no error, >0 : error

2.18 GRIDRD8 – Read grid into a REAL*8 array

```
    FUNCTION GRIDRD8 (FILENM, NX, NY, Z,
   .                  XMIN, XMAX, YMIN, YMAX, ZMIN, ZMAX)
    CHARACTER FILENM*(*)
    INTEGER*4 NX, NY, GRIDRD8
    REAL*8    Z(*), XMIN, XMAX, YMIN, YMAX, ZMIN, ZMAX
```

This routine reads the contents of a grid file into a REAL*8 array.
This array specified in the calling routine may be one- or two-dimensional.
The values will be stored row by row, starting at the lower left corner,
and pertain to the nodes of the grid. The coordinates of the value
stored in Z(KX,KY) will thus be:

```
  X = XMIN + (KX-1) * DX   with   DX = (XMAX-XMIN) / (NX-1)
  Y = YMIN + (KY-1) * DY   with   DY = (YMAX-YMIN) / (NY-1)
```

were NX and NY are the actual integer dimensions of the grid. The area
covered by the grid is always rectangular and has corners (XMIN,YMIN) and
(XMAX,YMAX).
If the array is specified one-dimensional in the calling routine, you may
specify an original X-dimension of 0 (NX=0) and NY equal to the maximum
size of the single dimension. In that case, the array will be filled as
were the original dimensions of the array equal to the actual dimensions
of the grid.
Points that do not have a defined value in the grid are given a value of
1.0D+35.

Arguments:

FILENM	(input):	Name of the file containing the grid.
NX	(input):	X-dimension of array Z as defined in the calling routine. If NX=0, Z is defined one-dimensional.
	(output):	Actual X-dimension of the grid.
NY	(input):	Y-dimension of array Z as defined in the calling routine. If NX=0, NY specifies the maximum single dimension.
	(output):	Actual Y-dimension of the grid, or zero if error occurred.
Z	(output):	Array into which the gridded data must be stored.
XMIN	(output):	X (longitude) value of lower left corner.
XMAX	(output):	X (longitude) value of upper right corner.
YMIN	(output):	Y (latitude) value of lower left corner.
YMAX	(output):	Y (latitude) value of upper right corner.
ZMIN	(output):	minimum Z (height) value in the grid.
ZMAX	(output):	maximum Z (height) value in the grid.
GRIDRD8	(output):	Return code: =0 : no error, >0 : error

2.19 GRIDWR4 – Write REAL*4 array to a file

```
    FUNCTION GRIDWR4 (FILENM, NX, NY, Z, MX, XMIN, XMAX, YMIN, YMAX)
    INTEGER*4 NX, NY, MX, GRIDWR4
    REAL*4    Z(MX,*), XMIN, XMAX, YMIN, YMAX
    CHARACTER FILENM*(*)
```

```
This routine writes the contents of a REAL*4 array into a grid file.
This array specified in the calling routine may be one- or two-dimensional.
The values will be stored row by row, starting at the lower left corner,
and pertain to the nodes of the grid. The coordinates of the value
stored in Z(KX,KY) will thus be:

  X = XMIN + (KX-1) * DX   with   DX = (XMAX-XMIN) / (NX-1)
  Y = YMIN + (KY-1) * DY   with   DY = (YMAX-YMIN) / (NY-1)

were NX and NY are the actual integer dimensions of the grid. The area
covered by the grid is always rectangular and has corners (XMIN,YMIN) and
(XMAX,YMAX).
If the array is specified one-dimensional in the calling routine, you should
specify MX=NX in the call to GRIDWR4.
If values in the array are larger than 1D20, they will be labeled 'undefined'
in the grid. Upon the next GRIDRD, a value of 1D35 will be returned for these
points.

Arguments:
 FILENM   (input): Name of the file containing the grid.
 NX       (input): Actual X-dimension of the grid.
 NY       (input): Actual Y-dimension of the grid.
 Z        (input): Array in which the gridded data is stored.
 MX       (input): X-dimension of the array as defined by the calling
                   routine or program.
 XMIN     (input): X (longitude) value of lower left corner.
 XMAX     (input): X (longitude) value of upper right corner.
 YMIN     (input): Y (latitude) value of lower left corner.
 YMAX     (input): Y (latitude) value of upper right corner.
 GRIDWR4 (output): Return code:
                   =0 : normal return, >0 : error occurred
```

2.20 GRIDWR8 – Write REAL*8 array to a file

```
    FUNCTION GRIDWR8 (FILENM, NX, NY, Z, MX, XMIN, XMAX, YMIN, YMAX)
    INTEGER*4 NX, NY, MX, GRIDWR8
    REAL*8    Z(MX,*), XMIN, XMAX, YMIN, YMAX
    CHARACTER FILENM*(*)
```

```
This routine writes the contents of a REAL*8 array into a grid file.
This array specified in the calling routine may be one- or two-dimensional.
The values will be stored row by row, starting at the lower left corner,
and pertain to the nodes of the grid. The coordinates of the value
stored in Z(KX,KY) will thus be:

  X = XMIN + (KX-1) * DX   with   DX = (XMAX-XMIN) / (NX-1)
  Y = YMIN + (KY-1) * DY   with   DY = (YMAX-YMIN) / (NY-1)

were NX and NY are the actual integer dimensions of the grid. The area
covered by the grid is always rectangular and has corners (XMIN,YMIN) and
(XMAX,YMAX).
If the array is specified one-dimensional in the calling routine, you should
specify MX=NX in the call to GRIDWR8.
If values in the array are larger than 1D20, they will be labeled 'undefined'
in the grid. Upon the next GRIDRD, a value of 1D35 will be returned for these
points.

Arguments:
 FILENM   (input): Name of the file containing the grid.
 NX       (input): Actual X-dimension of the grid.
 NY       (input): Actual Y-dimension of the grid.
 Z        (input): Array in which the gridded data is stored.
 MX       (input): X-dimension of the array as defined by the calling
                   routine or program.
 XMIN     (input): X (longitude) value of lower left corner.
 XMAX     (input): X (longitude) value of upper right corner.
 YMIN     (input): Y (latitude) value of lower left corner.
 YMAX     (input): Y (latitude) value of upper right corner.
 GRIDWR8 (output): Return code:
                   =0 : normal return, >0 : error occurred
```

2.21 GRILLU – 'Illuminate' geometry grid

```
    SUBROUTINE GRILLU (H, G, ANGLE, NX, NY, MX)
    INTEGER NX, NY, MX
    REAL H(MX,*), G(MX,*), ANGLE

Illuminate a geometry grid H in direction ANGLE and obtain grid G

Arguments:
 H     (input): Geometry grid
 G    (output): Illumination grid = derivative of H along a line
                which makes an angle ANGLE with the 'east-west' axis
 ANGLE (input): Angle of illumination (in degrees), 0 is in 'east-west'
                direction, 90 is in a 'north-south' direction
 NX,NY (input): Dimension of the grid
 MX    (input): First dimension of arrays H and G
```

2.22 GRSTAT4 – Determine grid statistics

```
    SUBROUTINE GRSTAT4 (GRID, NX, NY, MX, YMIN, YMAX, FACT,
   |           NVALID, ZMIN, ZMAX, ZMEAN, ZRMS, ZSIGMA)
    INTEGER*4 NX, NY, MX, NVALID
    REAL*4 GRID(MX,*), YMIN, YMAX, FACT,
   | ZMIN, ZMAX, ZMEAN, ZRMS, ZSIGMA
```

Subroutine to compute statistics of values contained on a grid.
Weighting by latitude is invoked by specifying YMIN and YMAX as the
minimum and maximum latitude of the grid. Weighting is disabled when
both are set to zero.

The routine also allow an iterative editing of outliers, which are off
by more than FACT times the standard deviation from the mean. Use
a non-positive value of FACT to disable editing.

Upon return, the values ZMIN, ZMAX, ZMEAN, ZSIGMA are set to the minimum,
maximum, mean, and standard deviation of the NVALID values in the grid
(after editing).

Input arguments:

GRID	: Grid of dimension NX*NY. The first dimension of the array as defined in the calling (sub)program is MX.
NX, NY	: Number of grid values in X and Y direction.
MX	: Dimension of array GRID as defined in the calling (sub)program.
YMIN,YMAX	: Latitude boudaries of the grid in degrees. To disable weighting by the cosine of latitude, specify 0 for both values.
FACT	: Factor determines the editing range: [ZMEAN - FACT*ZSIGMA ; ZMEAN + FACT*ZSIGMA]

Output arguments:

NVALID	: Number of non-edited grid values.
ZMIN,ZMAX	: Minimum and maximum value in the grid.
ZMEAN	: (Weighted) mean of the grid values.
ZRMS	: (Weighted) RMS of the grid values.
ZSIGMA	: (Weighted) standard deviation of the grid values.

2.23 GRSTAT8 – Determine grid statistics

```
    SUBROUTINE GRSTAT8 (GRID, NX, NY, MX, YMIN, YMAX, FACT,
   |           NVALID, ZMIN, ZMAX, ZMEAN, ZRMS, ZSIGMA)
    INTEGER*4 NX, NY, MX, NVALID
    REAL*8 GRID(MX,*), YMIN, YMAX, FACT,
   | ZMIN, ZMAX, ZMEAN, ZRMS, ZSIGMA
```

Subroutine to compute statistics of values contained on a grid.
Weighting by latitude is invoked by specifying YMIN and YMAX as the
minimum and maximum latitude of the grid. Weighting is disabled when
both are set to zero.

The routine also allow an iterative editing of outliers, which are off
by more than FACT times the standard deviation from the mean. Use
a non-positive value of FACT to disable editing.

Upon return, the values ZMIN, ZMAX, ZMEAN, ZSIGMA are set to the minimum,
maximum, mean, and standard deviation of the NVALID values in the grid
(after editing).

Input arguments:

GRID	: Grid of dimension NX*NY. The first dimension of the array as defined in the calling (sub)program is MX.
NX, NY	: Number of grid values in X and Y direction.
MX	: Dimension of array GRID as defined in the calling (sub)program.
YMIN,YMAX	: Latitude boudaries of the grid in degrees. To disable weighting by the cosine of latitude, specify 0 for both values.
FACT	: Factor determines the editing range: [ZMEAN - FACT*ZSIGMA ; ZMEAN + FACT*ZSIGMA]

Output arguments:

NVALID	: Number of non-edited grid values.
ZMIN,ZMAX	: Minimum and maximum value in the grid.
ZMEAN	: (Weighted) mean of the grid values.
ZRMS	: (Weighted) RMS of the grid values.
ZSIGMA	: (Weighted) standard deviation of the grid values.

2.24 GRWINT4 – Interpolate a two-dimensional array using a weight function

```
FUNCTION GRWINT4 (GRID, X, Y, NX, NY, MX, SIGMA, CUTOFF)
INTEGER*4 NX, NY, MX
REAL*4      GRWINT4, GRID(MX,*), X, Y, SIGMA, CUTOFF
```

This subroutine interpolates a two-dimensional regular grid using the exponential weight function

```
    W = EXP ( -(D/SIGMA)**2 )
```

where D is the distance from the point to be interpolated to any of the neighbouring grid points, SIGMA is a smoothing factor. The number of points that are considered 'neighbouring' is determined by the argument CUTOFF, which is the minimum weight of grid points to be used in the interpolation.
The dimension of the grid is NX by NY. The real coordinates of the point to be interpolated are (X,Y) where 1 <= X <= NX and 1 <= Y <= NY.

Arguments:

GRID	(input):	Two dimension array containing grid.
X,Y	(input):	REAL*4 grid coordinates of the point to be interpolated.
NX,NY	(input):	X- and Y- dimension of the grid.
MX	(input):	First dimension of the array as defined in the calling routine. MX >= NX.
SIGMA	(input):	Smoothing factor (as given above) in grid distances.
CUTOFF	(input):	Minimum weight of points to be included in the weighting.
GRWINT4	(output):	Interpolated value at point (X,Y).

2.25 GRWINT8 – Interpolate a two-dimensional array using a weight function

```
    FUNCTION GRWINT8 (GRID, X, Y, NX, NY, MX, SIGMA, CUTOFF)
    INTEGER*4 NX, NY, MX
    REAL*8      GRWINT8, GRID(MX,*), X, Y, SIGMA, CUTOFF

This subroutine interpolates a two-dimensional regular grid using the
exponential weight function
    W = EXP ( -(D/SIGMA)**2 )
where D is the distance from the point to be interpolated to any of
the neighbouring grid points, SIGMA is a smoothing factor.
The number of points that are considered 'neighbouring' is determined
by the argument CUTOFF, which is the minimum weight of grid points to be
used in the interpolation.
The dimension of the grid is NX by NY. The real coordinates of the point to be
interpolated are (X,Y) where 1 <= X <= NX and 1 <= Y <= NY.

Arguments:
GRID    (input): Two dimension array containing grid.
X,Y     (input): REAL*8 grid coordinates of the point to be interpolated.
NX,NY   (input): X- and Y- dimension of the grid.
MX      (input): First dimension of the array as defined in the calling
                 routine. MX >= NX.
SIGMA   (input): Smoothing factor (as given above) in grid distances.
CUTOFF  (input): Minimum weight of points to be included in the weighting.
GRWINT8 (output): Interpolated value at point (X,Y).
```

2.26 MASKBINF – Return information about buffered mask

```
    SUBROUTINE MASKBINF (POINTER, NX, NY, XMIN, XMAX, YMIN, YMAX)
    INTEGER*4 POINTER, NX, NY
    REAL*8    XMIN, XMAX, YMIN, YMAX

This routine returns information about a mask previously loaded with
the subroutine MASKBUFF.

Input argument:
 POINTER    : Pointer to grid structure as returned by GRIDBUFF

Output arguments:
 NX, NY     : Number of maskcells in X- and Y-direction
 XMIN, XMAX : X-coordinate of the outer limits of
              left- and rightmost maskcell
 YMIN, YMAX : Y-coordinate of the outer limits of
              lower- and uppermost maskcell
```

2.27 MASKBINT – Interogate mask in buffer

```
    FUNCTION MASKBINT (POINTER, X, Y)
    INTEGER  MASKBINT, POINTER
    REAL*8   X, Y

This function determines whether the mask bit at location (X,Y)
is set or not. The mask is loaded by MASKBUFF and pointed to by
POINTER.

The location at which the mask is interogated is given by X and
Y. These arguments are given in "world coordinates"; in other words, X
must be between XMIN and XMAX, the coordinates of the left- and
right-most mask point. Something similar holds for the Y-coordinate.

Upon exit, the function value MASKBINT will be either 1 or 0,
depending on the value of the mask at the pixel in which (X, Y)
resides.

When X and/or Y are out of the limits of the mask a value -1 is
returned.

Input arguments:
 POINTER  : Pointer to the mask structure as returned by GRIDBUFF
 X, Y     : X- and Y-coordinate of the point to be interpolated

Output argument:
 MASKBINT : Mask bit at the location (X, Y)
            0 = bit was not set
            1 = bit was set
           -1 = error occurred (e.g. X or Y out of range)
```

2.28 MASKBUFF – Load a mask file into memory

```
    FUNCTION MASKBUFF (FILENM, POINTER)
    INTEGER*4 MASKBUFF, POINTER
    CHARACTER*(*) FILENM

This routine allocated memory and loads the contents of a mask
file into the allocated memory.

The mask file has to be coded in the PBM format. In the
header of the PBM file the mask area can be coded using a line
# AREA <lon0> <lon1> <lat0> <lat1>
If this line is absent the default will be assumed:
# AREA 0 360 -90 90

The routine MASKBINT can be used to interogate the mask buffer
to determine the flag bit at a given latitude and longitude.

Other information on the mask, like mask dimension, can be obtained
with the MASKBINF routine.

When the allocation of memory or the loading of the grid was
unsuccessful, this will be reflected in the returned pointer value.

Input argument:
 FILENM   : Name of the BPM file containing the mask.

Output argument:
 POINTER  : Pointer to the mask structure

Return value:
 MASKBUFF : 0 = No error
            1 = Could not find or open mask
            2 = Ilegal mask format
            3 = Memory allocation was unsuccessful
            4 = Error loading mask
```

2.29 SGRIDRD4 – Read grid into a REAL*4 array

```
    FUNCTION SGRIDRD4 (FILENM, NX, NY, Z,
   .                   XMIN, XMAX, YMIN, YMAX, ZMIN, ZMAX)
    CHARACTER FILENM*(*)
    INTEGER*4 NX, NY, SGRIDRD4
    REAL*4    Z(*), XMIN, XMAX, YMIN, YMAX, ZMIN, ZMAX

This routine reads the contents of a grid file into a REAL*4 array.
This array specified in the calling routine may be one- or two-dimensional.
The values will be stored row by row, starting at the lower left corner,
and pertain to the nodes of the grid. The coordinates of the value
stored in Z(KX,KY) will thus be:

  X = XMIN + (KX-1) * DX   with   DX = (XMAX-XMIN) / (NX-1)
```

```
  Y = YMIN + (KY-1) * DY   with   DY = (YMAX-YMIN) / (NY-1)

were NX and NY are the actual integer dimensions of the grid. The area
covered by the grid is always rectangular and has corners (XMIN,YMIN) and
(XMAX,YMAX).
If the array is specified one-dimensional in the calling routine, you may
specify an original X-dimension of 0 (NX=0) and NY equal to the maximum
size of the single dimension. In that case, the array will be filled as
were the original dimensions of the array equal to the actual dimensions
of the grid.
Points that do not have a defined value in the grid are given a value of
1.0D+35.

Subgrids.
If the grid boudaries are not given a priori (i.e. when XMIN=XMAX or
YMIN=YMAX), then the entire grid as it is stored in returned, along with
its actual boundaries. If the boundaries are predefined, a subsection of
the grid is selected. Still, the actual boundaries are returned, which
may differ from the a priori because the grid mesh may not comply with
the selected boundaries.

Sampling.
If the array size declared in the calling routine (defined by NX and NY)
is too small to contain the grid (or the requested subsection), the
grid is sampled. Only one out of 1, 2, 3, ... grid points in each
direction will be returned.

Arguments:
 FILENM   (input): Name of the file containing the grid.
 NX       (input): X-dimension of array Z as defined in the calling
                   routine. If NX=0, Z is defined one-dimensional.
         (output): Actual X-dimension of the grid.
 NY       (input): Y-dimension of array Z as defined in the calling routine.
                   If NX=0, NY specifies the maximum single dimension.
         (output): Actual Y-dimension of the grid, or zero if error occurred.
 Z       (output): Array into which the gridded data must be stored.
 XMIN    (in/output): X (longitude) value of lower left corner.
 XMAX    (in/output): X (longitude) value of upper right corner.
 YMIN    (in/output): Y (latitude) value of lower left corner.
 YMAX    (in/output): Y (latitude) value of upper right corner.
 ZMIN    (output): minimum Z (height) value in the grid.
 ZMAX    (output): maximum Z (height) value in the grid.
 SGRIDRD4 (output): Return code:
                   =0 : no error, >0 : error
```

2.30 SGRIDRD8 – Read grid into a REAL*8 array

```
    FUNCTION SGRIDRD8 (FILENM, NX, NY, Z,
   .                   XMIN, XMAX, YMIN, YMAX, ZMIN, ZMAX)
    CHARACTER FILENM*(*)
    INTEGER*4 NX, NY, SGRIDRD8
    REAL*8    Z(*), XMIN, XMAX, YMIN, YMAX, ZMIN, ZMAX

This routine reads the contents of a grid file into a REAL*8 array.
```

```
This array specified in the calling routine may be one- or two-dimensional.
The values will be stored row by row, starting at the lower left corner,
and pertain to the nodes of the grid. The coordinates of the value
stored in Z(KX,KY) will thus be:

  X = XMIN + (KX-1) * DX   with   DX = (XMAX-XMIN) / (NX-1)
  Y = YMIN + (KY-1) * DY   with   DY = (YMAX-YMIN) / (NY-1)

were NX and NY are the actual integer dimensions of the grid. The area
covered by the grid is always rectangular and has corners (XMIN,YMIN) and
(XMAX,YMAX).
If the array is specified one-dimensional in the calling routine, you may
specify an original X-dimension of 0 (NX=0) and NY equal to the maximum
size of the single dimension. In that case, the array will be filled as
were the original dimensions of the array equal to the actual dimensions
of the grid.
Points that do not have a defined value in the grid are given a value of
1.0D+35.

Subgrids.
If the grid boudaries are not given a priori (i.e. when XMIN=XMAX or
YMIN=YMAX), then the entire grid as it is stored in returned, along with
its actual boundaries. If the boundaries are predefined, a subsection of
the grid is selected. Still, the actual boundaries are returned, which
may differ from the a priori because the grid mesh may not comply with
the selected boundaries.

Sampling.
If the array size declared in the calling routine (defined by NX and NY)
is too small to contain the grid (or the requested subsection), the
grid is sampled. Only one out of 1, 2, 3, ... grid points in each
direction will be returned.

Arguments:
 FILENM   (input): Name of the file containing the grid.
 NX       (input): X-dimension of array Z as defined in the calling
                   routine. If NX=0, Z is defined one-dimensional.
         (output): Actual X-dimension of the grid.
 NY       (input): Y-dimension of array Z as defined in the calling routine.
                   If NX=0, NY specifies the maximum single dimension.
         (output): Actual Y-dimension of the grid, or zero if error occurred.
 Z       (output): Array into which the gridded data must be stored.
 XMIN    (in/output): X (longitude) value of lower left corner.
 XMAX    (in/output): X (longitude) value of upper right corner.
 YMIN    (in/output): Y (latitude) value of lower left corner.
 YMAX    (in/output): Y (latitude) value of upper right corner.
 ZMIN    (output): minimum Z (height) value in the grid.
 ZMAX    (output): maximum Z (height) value in the grid.
 SGRIDRD8 (output): Return code:
                   =0 : no error, >0 : error
```