MEMORANDUM
RESEARCH DEPARTMENT

From: *Mats Hamrud* ***Date:*** *Jun 27, 2002*

To: RD Scientific Staff and Consultants. ***File:*** R48.3/MH/4

Copy: HR, HO, HMD, HMAS, HMOS, O.Pesonen, G.Austad, Jean Pailleux, Jean-François Geleyn,UGAMP (for info.).

Subject: New Transform Library

As from cycle 23r4 the spectral transforms have been broken out of the IFS to form a separate library (libtrans), the source residing in its own VOB (trans). There are several reasons for this change. One is to make the IFS more modular, the transforms form a non-scientific part that could easily be separated from the rest. Another reason is to make the efficient spectral transforms previously buried within the IFS usable by other codes.

The routines in the transform package are divided into two groups, externally callable routines and internal routines. The internal routines are all Fortran 90 module procedures and are not described in this documentation. The externally callable routines are not module procedures but an explicit interface block is needed in order to call them. This makes it possible to use Fortran 90 features like assumed shape arrays and optional arguments. The interface blocks can all be found in the trans VOB (in the interface directory).

The assumed shape arrays are used to avoid having to pass dimensions and to ensure that arrays are dimensioned correctly. When calling a transform routine it always transforms the whole of the arrays passed, e.g. CALL INV_TRANS(PSPSCALAR=SPEC,PGP=GP) would transform all the fields of SPEC into GP. The spectral array passed into the transform routines must be of rank 2, the first dimension for the number of fields and the second for the spectral coefficients.

In the following description of the individual routines [..] denotes default values for optional input arguments. Where arguments are arrays (:,...) is used to show the rank of the array. Following IFS (Doctor) coding conventions argument names starting with K denotes integer arguments, starting letter P indicates real arguments and L logical arguments. Arguments names in italics indicate that the argument in question is only of interest in the case of using more than one processor. When the word GLOBAL is used in the following documentation it refers to viewing the data as a whole, not the part of it that is available on an individual processor in the distributed case.

After the documentation of the routines follows some simple examples.

SETUP_TRANS0 - General setup routine for transform package

Purpose.

Resolution independent part of setup of transform package Has to be called before SETUP_TRANS.

Interface.

CALL SETUP_TRANS0(...)

No required input arguments and no output arguments.

Optional input arguments.

[..] indicates default value.

```
KOUT         - Unit number for listing output [6]
KERR         - Unit number for error messages [0]
KPRINTLEV    - Level of output to KOUT, 0->no output,1->normal,2->debug [0]
KMAX_RESOL   - Maximum number of different resolutions for this run [1]
KPRGPNS      - Number of processors in N-S direction in grid-point space [1]
KPRGPEW      - Number of processors in E-W direction in grid-point space [1]
KPRTRW       - Number of processors in wave direction in spectral space [1]
LDALLOPERM   - Allocate some arrays permanently (OpenMP issue) [false]
```

The total number of (MPI)-processors has to be equal to KPRGPNS*KPRGPEW

SETUP_TRANS - Setup transform package for specific resolution

Purpose.

To setup for making spectral transforms. Each call to this routine creates a new resolution up to a maximum of KMAX_RESOL as set up in SETUP_TRANS0. You need to call SETUP_TRANS0 before this routine can be called. The optional parameter KRESOL used in subsequent calls to other transform routines refers to the n'th defined resolution.

Interface.

CALL SETUP_TRANS(...)

No output arguments.

Required input arguments:

```
KSMAX       - Spectral truncation required
KDGL        - Number of Gaussian latitudes
```

Optional input arguments:

```
KLOEN(:)     - Number of points on each Gaussian latitude [2*KDGL]
LDSPLIT      - True if split latitudes in grid-point space [false]
KAPSETS      - Number of apple sets in the distribution [0]
KTMAX        - Truncation order for tendencies [KSMAX]
```

KSMAX,KDGL,KTMAX and KLOEN are GLOBAL variables describing the resolution in spectral and grid-point space

LDSPLIT and KAPSETS describe the distribution among processors of grid-point data and have no relevance if you are using a single processor

DIR_TRANS - Direct spectral transform (from Gaussian grid to spectral space).

Purpose.

Interface routine for the Direct Spectral Transform

Interface.

CALL DIR_TRANS(....)

Required input arguments:

In the following description NF_UV_G is the GLOBAL number of u/v type fields and NF_SCALARS_G is the GLOBAL number of scalarvalued fields. When the fields are not distributed over processors, NF_UV_G is given by the length of PSPVOR and PSPDIV and NF_SCALARS_G by the length of PSPSCALAR. If the fields are distributed over processors (the case where KPRTRW < KPRGPNS*KPRGPEW in SETUP_TRANS0), the arguments KVSETUV and KVSETSC describing the distribution have to be present and their respective lengths give NF_UV_G and/or NF_SCALARS_G.

```
PGP(:,:,:)     - gridpoint fields
```

PGP must be dimensioned (KPROMA,NF_GP,NGPBLKS) where KPROMA is the blocking factor(see below), NF_GP the total number of fields in gridpoint space and NGPBLKS the number of KPROMA blocks. The default for KPROMA is the total number of gridpoints on a processor in which case NGPBLKS is 1. The ordering of the output fields is as follows (all parts are optional depending on the input switches):

u : NF_UV_G fields (if psvor and pspdiv present)
v : NF_UV_G fields (if psvor and pspdiv present)
scalar fields : NF_SCALARS_G fields (if pspscalar present)

Optional input arguments:

KRESOL - resolution identifier which is to be used [first defined resolution]
KPROMA - requested blocking factor for gridpoint output. Used to divide the gridpoint array in chunks of a size suitable for further computations (to control memory usage, vectorization etc.) [total number of gridpoints on processor (KGPTOT from TRANS_INQ)]
KVSETUV(:) - indicating which 'field-set' in spectral space owns a vor/div field. Equivalent to NBSETLEV in the IFS. The length of KVSETUV should be the GLOBAL number of u/v fields which is the dimension of u and v related fields in grid-point space.
KVESETSC(:) - indicating which 'field-set' in spectral space owns a scalar field. As for KVSETUV this argument is required if the total number of processors is greater than the number of processors used for distribution in spectral wave space.

Required output arguments :

None but either PSPVOR and PSPDIV or PSPSCALAR has to be present(see below).

Optional output arguments:

PSPVOR(:,:) - spectral vorticity
PSPDIV(:,:) - spectral divergence
PSPSCALAR(:,:) - spectral scalarvalued fields

Here the first dimension is the field dimension and the second is for the spectral coefficents. In the case of one processor the ordering of the spectral coefficents is the same as the one obtained when decoding/encoding a GRIB field using the GRIBEX routine.

DIR_TRANSAD - Adjoint of Direct spectral transform .

See DIR_TRANS. Only change is that PGP becomes output argument and PSPVOR,PSPDIV and PSPSCALAR input arguments.

INV_TRANS - Inverse spectral transform (spectral to gridpoint).

Purpose.

Interface routine for the inverse spectral transform. Also for computing gridpoint u and v from vorticity and divergence and for computing N-S and E-W derivatives of fields.

Interface.

CALL INV_TRANS(...)

In the following description NF_UV_G is the GLOBAL number of u/v type fields and NF_SCALARS_G is the GLOBAL number of scalarvalued fields. When the fields are not distributed over processors, NF_UV_G is given by the length of PSPVOR and PSPDIV and

NF_SCALARS_G by the length of PSPSCALAR. If the fields are distributed over processors (the case where KPRTRW < KPRGPNS*KPRGPEW in SETUP_TRANS0), the arguments KVSETUV and KVSETSC describing the distribution have to be present and their respective lengths give NF_UV_G and/or NF_SCALARS_G.

Input required arguments:

None.

Input optional arguments.

PSPVOR(:,:) – spectral vorticity
PSPDIV(:,:) – spectral divergence
PSPSCALAR(:,:) – spectral scalarvalued fields
FSPGL_PROC – external procedure to be executed in fourier space before transposition [none]
LDSCDERS – indicating if derivatives of scalar variables are req. [false]
LDVORGP – indicating if grid-point vorticity is req. [false]
LDDIVGP – indicating if grid-point divergence is req. [false]
LDUVDER – indicating if E-W derivatives of u and v are req. [false]
KPROMA – required blocking factor for gridpoint output. Used to divide the gridpoint array in chunks of a size suitable for further computations (to control memory usage, vectorization etc.) [total number of gridpoints for one field]
KVSETUV(:) – indicating which 'field-set' in spectral space owns a vor/div field. Equivalant to NBSETLEV in the IFS. The length of KVSETUV should be the GLOBAL number of u/v fields which is the dimension of u and v releated fields in grid-point space.
KVESETSC(:) – indicating which 'field-set' in spectral space owns a scalar field. As for KVSETUV this argument is required if the total number of processors is greater than the number of processors used for distribution in spectral wave space.
KRESOL – resolution identifier which is required [first defined resolution]

Output required arguments.

PGP(:,:,:) – gridpoint fields

PGP need to dimensioned (KPROMA,NF_GP,NGPBLKS) where KPROMA is the blocking factor, NF_GP the total number of output fields and NGPBLKS the number of KPROMA blocks. The default for KPROMA is the total number of points (NGPTOT) in which case NGPBLKS is 1.

The ordering of the output fields is as follows (all parts are optional depending on the input switches):

```
vorticity      : NF_UV_G fields (if psvor/pspdiv present and LDVORGP)
divergence     : NF_UV_G fields (if psvor/pspdiv present and LDDIVGP)
u              : NF_UV_G fields (if psvor/pspdiv present)
v              : NF_UV_G fields (if psvor/pspdiv present)
scalar fields : NF_SCALARS_G fields (if pspscalar present)
N-S derivative of scalar fields : NF_SCALARS_G fields (if pspscalar present
and LDSCDERS)
E-W derivative of u : NF_UV_G fields (if psvor/pspdiv present and and LDUVDER)
E-W derivative of v : NF_UV_G fields (if psvor/pspdiv present and and LDUVDER)
E-W derivative of scalar fields : NF_SCALARS_G fields (if pspscalar present
and LDSCDERS)
```

Output optional arguments:

None

INV_TRANSAD - Adjoint of Inverse spectral transform (spectral to gridpoint).

See INV_TRANS. Only difference is that PGP becomes an input argument and PSPVOR,PSPDIV and PSPSCALAR becomes output arguments.

TRANS_END - Terminate transform package

Purpose.

Terminate transform package. Release all allocated arrays.

Interface.

CALL TRANS_END

Arguments :

None

TRANS_INQ - Extract information from the transform package

Purpose.

Interface routine for extracting information from the Transform Package.

Interface.

CALL TRANS_INQ(...)

No required arguments.

Optional input arguments :

KRESOL – resolution identifier for which info is required [first defined resolution]

Optional output arguments:

SPECTRAL SPACE

KSPEC – Number of complex spectral coefficients on this PE
KSPEC2 – 2*KSPEC (for use as second dimension of PSPVOR etc.)
KSPEC2G – Global KSPEC2
KSPEC2MX – Maximum KSPEC2 among all PEs
KNUMP – Number of spectral waves handled by this PE
KGPTOT – Total number of grid columns on this PE
KGPTOTG – Total number of grid columns on the Globe
KGPTOTMX – Maximum number of grid columns on any of the PEs
KMYMS(:) – This PEs spectral zonal wavenumbers
KASM0(0:) – Address in a spectral array of (m, n=m)
KUMPP(:) – No. of wave numbers each wave set is responsible for
KPOSSP(:) – Defines partitioning of global spectral fields among PEs
KPTRMS(:) – Pointer to the first wave number of a given a-set
KALLMS(:) – Wave numbers for all wave-set concatenated together to give all wave numbers in wave-set order
KDIM0G(0:) – Defines partitioning of global spectral fields among PEs

GRIDPOINT SPACE

KFRSTLAT(:) – First latitude of each a-set in grid-point space
KLSTTLAT(:) – Last latitude of each a-set in grid-point space
KFRSTLOFF – Offset for first lat of own a-set in grid-point space
KPTRLAT(:) – Pointer to the start of each latitude
KPTRFRSTLAT(:) – Pointer to the first latitude of each a-set in NSTA and NONL arrays
KPTRLSTLAT(:) – Pointer to the last latitude of each a-set in NSTA and NONL arrays
KPTRFLOFF – Offset for pointer to the first latitude of own a-set NSTA and NONL arrays, i.e. nptrfrstlat(myseta)-1
KSTA(:,:) – Position of first grid column for the latitudes on a processor. The information is available for all processors. The b-sets are distinguished by the last dimension of nsta().The latitude band for each a-set is addressed by nptrfrstlat(jaset),nptrlstlat(jaset), and nptrfloff=nptrfrstlat(myseta) on this processors a-set. Each split latitude has two entries in nsta(,:) which necessitates the rather complex addressing of nsta(,:) and the overdimensioning of nsta by nprgpns. See IFS documentaion (Appendix A) for more details.
KONL(:,:) – Number of grid columns for the latitudes on a processor. Similar to nsta() in data structure.
LDSPLITLAT(:) – TRUE if latitude is split in grid point space over two a-sets

FOURIER SPACE

KULTPP(:) – Number of latitudes for which each a-set is calculating the FFT’s.
KPTRLS(:) – Pointer to first global latitude of each a-set for which it performs the Fourier calculations

LEGENDRE POLYNOMIALS

```
PMU(:)         - sin(Gaussian latitudes)
PGW(:)         - Gaussian weights
PRPNM(:,:)     - Legendre polynomials on this PE
```

EXAMPLES.

Both the following examples are for running on one processor only. For more complex examples see the IFS code (CY23R4), routines SUTRANS, SUMP, TRANSINV_MDL, TRANSDIR_MDL etc.

To compile a program similar to these examples you need to have the directory containing the interface blocks (/cc/rd/trans/interface) visable with a view corresponding to the version of the trans library you are using and this directory added to your search path for include files. To load you need to load with -ltrans -lifsaux -lmpi_serial (apart from your own libraries) with a suitable load path (e.g. /cc/rd/lib/23r4 on SGI workstations).

The following example transforms 10 spectral fields into grid-point. The routines INI_NLOEN, READSPEC and WRITEGRID are user routines (not shown).

```
PROGRAM EXAMPLE1
IMPLICIT NONE
INTEGER NLOEN(320),IFLDS
REAL,ALLOCATABLE :: SPEC(:,:),GP(:,:,:)
INTERFACE
#include "setup_trans0.h"
#include "setup_trans.h"
#include "trans_inq.h"
#include "inv_trans.h"
END INTERFACE
CALL INI_NLOEN(NLOEN) ! Initialize array descibing reduced grid
CALL SETUP_TRANS0
CALL SETUP_TRANS(KSMAX=319,KDGL=320,KLOEN=NLOEN)
CALL TRANS_INQ(KSPEC2=NSPEC2,KGPTOT=NGPTOT)
IFDLS=10
ALLOCATE(SPEC(IFLDS,NSPEC2))
CALL READSPEC(SPEC,IFLDS,NSPEC2) ! Read in spectral fields
ALLOCATE(GP(NGPTOT,IFLDS,1))
CALL INV_TRANS(PSPSCALAR=SPEC,PGP=GP)
CALL WRITEGRID(GP,IFLDS) ! Write out gridpint fields
END
```

This second example transforms grid-point u and v fields into spectral vorticity and divergence. Note the dimension of GP as 2*IFLDS to accomodate u followed by v. The routines INI_NLOEN, READGRID and WRITESPEC are user routines (not shown).

```
PROGRAM EXAMPLE2

IMPLICIT NONE
INTEGER NLOEN(320),IFLDS
REAL,ALLOCATABLE :: SPECVOR(:,:),SPECDIV,GP(:,:,:)
```

```
INTERFACE
#include “setup_trans0.h”
#include “setup_trans.h”
#include “trans_inq.h”
#include “dir_trans.h”
END INTERFACE

CALL INI_NLOEN(NLOEN) ! Initialize array descibing reduced grid
CALL SETUP_TRANS0
CALL SETUP_TRANS(KSMAX=319,KDGL=320,KLOEN=NLOEN)
CALL TRANS_INQ(KSPEC2=NSPEC2,KGPTOT=NGPTOT)
IFDLS=10
ALLOGATE(GP(NGPTOT,2*IFLDS,1))
CALL READGRID(GP,IFLDS) ! Read in u and v gridpint fields
ALLOCATE(SPECVOR(IFLDS,NSPEC2))
ALLOCATE(SPECDIV(IFLDS,NSPEC2))
CALL DIR_TRANS(PSPVOR=SPECVOR,PSPDIV=SPECDIV,PGP=GP)
CALL WRITESPEC(SPECVOR,SPECDIV,IFLDS,NSPEC2) ! Write out spectral fields
END
```