```
!-------------------------------------------------------------------------
subroutine trans_grid_to_spherical_3d(grid, spherical, do_truncation)
!-------------------------------------------------------------------------

real, intent(in), dimension (is:,:,:) :: grid
logical, intent(in), optional :: do_truncation
complex, intent(inout), dimension (ms:,ns:,:) :: spherical
real, dimension(num_lon,size(grid,2),size(grid,3)) :: grid_xglobal

logical :: do_truncation_local, grid_x_is_global
complex, dimension (0:num_lon/2,          js:je, size(grid,3)) :: fourier_g
complex, dimension (ms:me, je-js+1, size(grid,3), grid_layout(2)) :: fourier_s

integer(kind=kind(spherical)) :: c1, c2, c3

if(.not.module_is_initialized) then
  call error_mesg('trans_grid_to_spherical','transforms module is not initialized',
FATAL)
end if

if( size(grid,1).ne.ie-is+1 )&
     call mpp_error( FATAL, 'TRANSFORMS: size(grid,1).ne.ie-is+1.' )

if( size(grid,2).ne.je-js+1 )&
     call mpp_error( FATAL, 'TRANSFORMS: size(grid,2).ne.je-js+1.' )

if( size(spherical,1).ne.me-ms+1 )&
     call mpp_error( FATAL, 'TRANSFORMS: size(spherical,1).ne.me-ms+1.' )

if( size(spherical,2).ne.ne-ns+1 )&
     call mpp_error( FATAL, 'TRANSFORMS: size(spherical,2).ne.ne-ns+1.' )

if( size(spherical,3).ne.size(grid,3) )&
     call mpp_error( FATAL, 'TRANSFORMS: size(spherical,3).ne.size(grid,3).' )

if(present(do_truncation)) then
  do_truncation_local = do_truncation
else
  do_truncation_local = .true.
end if

call mpp_get_compute_domain( grid_domain, x_is_global=grid_x_is_global )
if( .NOT.grid_x_is_global )then
    grid_xglobal(is:ie,:,:) = grid(is:ie,:,:)
    call mpp_update_domains( grid_xglobal, grid_domain, XUPDATE )
    fourier_g = trans_grid_to_fourier(grid_xglobal)
else
    fourier_g = trans_grid_to_fourier(grid)
endif

if( trunc_fourier.lt.me )fourier_g(trunc_fourier+1:me,:,:) = cmplx(0.,0.)

call transpose_fourier( fourier_g, fourier_s )
call trans_fourier_to_spherical(fourier_s, spherical)

if(do_truncation_local) then
  if(triang_trunc_local) then
    call triangular_truncation(spherical)
  else
    call rhomboidal_truncation(spherical)
  end if
```

```
end if

if( debug )then
    c1 = mpp_chksum(spherical)
    c2 = mpp_chksum(fourier_g)
    c3 = mpp_chksum(grid)
    write(  0,'(a,i2,3z18,28i4)' )'G2S: ', mpp_pe(), c1, c2, c3, lbound(spherical),
ubound(spherical), &
         lbound(fourier_s), ubound(fourier_s), lbound(fourier_g), ubound(fourier_g),
lbound(grid), ubound(grid)
    call mpp_error( FATAL )
end if

return
end subroutine trans_grid_to_spherical_3d


!-------------------------------------------------------------------------------
 subroutine trans_spherical_to_grid_3d(spherical, grid)
!-------------------------------------------------------------------------------

complex, intent (in), dimension (ms:,ns:,:) :: spherical
real, intent(out), dimension (is:,:,:) :: grid
real, dimension(num_lon,size(grid,2),size(grid,3)) :: grid_xglobal

integer(kind=kind(spherical)) :: c1, c2, c3

complex, dimension (0:num_lon/2,          js:je, size(grid,3)) :: fourier_g
complex, dimension (ms:me, je-js+1, size(grid,3), grid_layout(2)) :: fourier_s
logical :: grid_x_is_global, spectral_y_is_global
type(domain1D) :: spectral_domain_y
integer, allocatable :: pelist(:)

if(.not.module_is_initialized) then
  call error_mesg('trans_spherical_to_grid','transforms module is not initialized',
FATAL)
end if

if( size(grid,1).ne.ie-is+1 )&
     call mpp_error( FATAL, 'TRANS_SPHERICAL_TO_GRID: size(grid,1).ne.ie-is+1.' )

if( size(grid,2).ne.je-js+1 )&
     call mpp_error( FATAL, 'TRANS_SPHERICAL_TO_GRID: size(grid,2).ne.je-js+1.' )

if( size(spherical,1).ne.me-ms+1 )&
     call mpp_error( FATAL, 'TRANS_SPHERICAL_TO_GRID: size(spherical,1).ne.me-ms+1.' )

if( size(spherical,2).ne.ne-ns+1 )&
     call mpp_error( FATAL, 'TRANS_SPHERICAL_TO_GRID: size(spherical,2).ne.ne-ns+1.' )

if( size(spherical,3).ne.size(grid,3) )&
     call mpp_error( FATAL, 'TRANS_SPHERICAL_TO_GRID: size(spherical,3).ne.size
(grid,3.' )

call trans_spherical_to_fourier( spherical, fourier_s )
call mpp_get_compute_domain( spectral_domain, y_is_global=spectral_y_is_global )
if( .NOT.spectral_y_is_global )then
    allocate( pelist(spectral_layout(2)) )
    call mpp_get_domain_components( spectral_domain, y=spectral_domain_y )
    call mpp_get_pelist( spectral_domain_y, pelist )
    call mpp_sum( fourier_s          , size(fourier_s(:,:,:,:)), pelist )
```

```
end if

call reverse_transpose_fourier( fourier_s, fourier_g )

fourier_g(trunc_fourier+1:num_lon/2,:,:) = cmplx(0.,0.)
call mpp_get_compute_domain( grid_domain, x_is_global=grid_x_is_global )
if( .NOT.grid_x_is_global )then
    grid_xglobal = trans_fourier_to_grid(fourier_g)
    grid(is:ie,:,:) = grid_xglobal(is:ie,:,:)
else
    grid = trans_fourier_to_grid(fourier_g)
endif

if( debug )then
    c1 = mpp_chksum(spherical)
    c2 = mpp_chksum(fourier_g)
    c3 = mpp_chksum(grid)
    write(  0,'(a,i2,3z18,28i4)' )'S2G: ', mpp_pe(), c1, c2, c3, lbound(spherical),
ubound(spherical), &
         lbound(fourier_s), ubound(fourier_s), lbound(fourier_g), ubound(fourier_g),
lbound(grid), ubound(grid)
end if

return
end subroutine trans_spherical_to_grid_3d

! Fields are transformed from the transform grid to spherical harmonics
!   and back by trans_spherical_to_grid and trans_grid_to_spherical
!
! The two utilities, divide_by_cos and divide_by_cos2, provide convenient
!   ways of dividing grid fields by cos(lat) and cos(lat)**2,
!   as is often required when transforming to and fro in different contexts,
!   without having to compute or retrieve the gaussian latitudes explicitly.
!
```