```
!-------------------------------------------------------------------------
subroutine compute_vor_div_3d(u_cos, v_cos, vorticity, divergence)
!-------------------------------------------------------------------------

complex, intent(in), dimension (:,:,:) :: u_cos
complex, intent(in), dimension (:,:,:) :: v_cos
complex, intent(out), dimension (:,:,:) :: vorticity
complex, intent(out), dimension (:,:,:) :: divergence

vorticity  = compute_alpha_operator(v_cos, u_cos, -1)
divergence = compute_alpha_operator(u_cos, v_cos, +1)

return
end subroutine compute_vor_div_3d


!-------------------------------------------------------------------------------
function compute_alpha_operator_3d(spherical_a, spherical_b, isign) result(alpha)
!-------------------------------------------------------------------------------

complex, intent(in), dimension (:,0:,:) :: spherical_a
complex, intent(in), dimension (:,0:,:) :: spherical_b
integer,intent(in) :: isign

complex, dimension (size(spherical_a,1), 0:size(spherical_a,2)-1, size
(spherical_a,3)) :: alpha

integer :: k

if(.not. module_is_initialized ) then
  call error_mesg('compute_vor or div','module spherical not initialized', FATAL)
end if

alpha = cmplx(0.,0.)

if( size(spherical_a,2).EQ.num_spherical+1 )then
!could be global domain, or only global in N
    if( size(spherical_a,1).EQ.num_fourier+1 )then
        do k=1,size(spherical_a,3)
           alpha(:,:,k) = coef_dx(:,:)*    &
                cmplx(-aimag(spherical_a(:,:,k)),real(spherical_a(:,:,k)))
           alpha(:,1:num_spherical,k) = alpha(:,1:num_spherical,k) -  &
                isign*coef_alpm(:,1:num_spherical)  &
                *spherical_b(:,0:num_spherical-1,k)
           alpha(:,0:num_spherical-1,k) = alpha(:,0:num_spherical-1,k) +  &
                isign*coef_alpp(:,0:num_spherical-1)*spherical_b(:,1:num_spherical,k)
        end do
    else if( size(spherical_a,1).EQ.me-ms+1 )then
        do k=1,size(spherical_a,3)
           alpha(:,:,k) = coef_dx(ms:me,:)*    &
                cmplx(-aimag(spherical_a(:,:,k)),real(spherical_a(:,:,k)))
           alpha(:,1:num_spherical,k) = alpha(:,1:num_spherical,k) -  &
                isign*coef_alpm(ms:me,1:num_spherical)  &
                *spherical_b(:,0:num_spherical-1,k)
           alpha(:,0:num_spherical-1,k) = alpha(:,0:num_spherical-1,k) +  &
                isign*coef_alpp(ms:me,0:num_spherical-1)*spherical_b
(:,1:num_spherical,k)
        end do
    endif
else if( size(spherical_a,1).EQ.me-ms+1 .AND. size(spherical_a,2).EQ.ne-ns+1 )then
!need to write stuff to acquire data at ns-1,ne+1
```

```
    call abort()
else
    call error_mesg( 'compute_alpha_operator_3d', 'invalid argument size', FATAL )
endif

return
end function compute_alpha_operator_3d
```