```
!-------------------------------------------------------------------
subroutine compute_ucos_vcos_3d(vorticity , divergence, u_cos, v_cos)
!-------------------------------------------------------------------

complex, intent(in), dimension (:,0:,:) :: vorticity
complex, intent(in), dimension (:,0:,:) :: divergence
complex, intent(out), dimension (:,0:,:) :: u_cos
complex, intent(out), dimension (:,0:,:) :: v_cos


integer :: k

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

if( size(vorticity,2).EQ.num_spherical+1 )then
!could be global domain, or only global in N
    if( size(vorticity,1).EQ.num_fourier+1 )then
        do k=1,size(vorticity,3)
           u_cos(:,:,k) = coef_uvc(:,:)*                                     &
                cmplx(-aimag(divergence(:,:,k)),real(divergence(:,:,k)))
           v_cos(:,:,k) = coef_uvc(:,:)*                                     &
                cmplx(-aimag(vorticity(:,:,k)),real(vorticity(:,:,k)))

           u_cos(:,1:num_spherical,k) = u_cos(:,1:num_spherical,k) +         &
                coef_uvm(:,1:num_spherical)*vorticity(:,0:num_spherical-1,k)
           v_cos(:,1:num_spherical,k) = v_cos(:,1:num_spherical,k) -         &
                coef_uvm(:,1:num_spherical)*divergence(:,0:num_spherical-1,k)

           u_cos(:,0:num_spherical-1,k) = u_cos(:,0:num_spherical-1,k) -     &
                coef_uvp(:,0:num_spherical-1)*vorticity(:,1:num_spherical,k)
           v_cos(:,0:num_spherical-1,k) = v_cos(:,0:num_spherical-1,k) +     &
                coef_uvp(:,0:num_spherical-1)*divergence(:,1:num_spherical,k)
        end do
    else if( size(vorticity,1).EQ.me-ms+1 )then
        do k=1,size(vorticity,3)
           u_cos(:,:,k) = coef_uvc(ms:me,:)*                                     &
                cmplx(-aimag(divergence(:,:,k)),real(divergence(:,:,k)))
           v_cos(:,:,k) = coef_uvc(ms:me,:)*                                     &
                cmplx(-aimag(vorticity(:,:,k)),real(vorticity(:,:,k)))

           u_cos(:,1:num_spherical,k) = u_cos(:,1:num_spherical,k) +         &
                coef_uvm(ms:me,1:num_spherical)*vorticity(:,0:num_spherical-1,k)
           v_cos(:,1:num_spherical,k) = v_cos(:,1:num_spherical,k) -         &
                coef_uvm(ms:me,1:num_spherical)*divergence(:,0:num_spherical-1,k)

           u_cos(:,0:num_spherical-1,k) = u_cos(:,0:num_spherical-1,k) -     &
                coef_uvp(ms:me,0:num_spherical-1)*vorticity(:,1:num_spherical,k)
           v_cos(:,0:num_spherical-1,k) = v_cos(:,0:num_spherical-1,k) +     &
                coef_uvp(ms:me,0:num_spherical-1)*divergence
(:,1:num_spherical,k)
        end do
    endif
else if( size(vorticity,1).EQ.me-ms+1 .AND. size(vorticity,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_ucos_vcos', 'invalid argument size', FATAL )
endif
```

```
return
end subroutine compute_ucos_vcos_3d

! Subroutine takes in vorticity and divergence fields and gives out u*cos(theta) and
v*cos(theta).
! From "Barotropic vorticity equation" documentation:
!       One can compute u and v from the vorticity in the spectral domain by first
computing the streamfunction (by simply dividing by the appropriate
!       eigenvalue of the Laplacian). Then v cos(θ) can be obtained immediately since a
zonal derivative consists of a simple multiplication by im in the
!       spectral domain. The meridional derivative needed to compute u can be obtained
with a recursion relation. More precisely, u m cos(θ) is a linear combination
!       of ψ_{l+1,m} and ψ_{l−1,m}. The underlying recursion relation is
```