```
#include "numbat_decl.h"

! Calculate the EM mode power using analytic expressions for the basis functions
.
!
! Sturmberg: Eq. (6)
!
!   P_z = 2 Re[\zhat \dot \int dx dy E^* \cross H] =  2 Re[ zhat \dot  \int dx d
y E_t^* \cross H_t]
!

subroutine em_mode_power_sz_analytic (k_0, n_modes, n_msh_el, n_msh_pts, &
   elnd_to_mesh, v_nd_xy, v_beta, soln_em_e, m_power, errco, emsg)


!     k_0 = 2 pi / lambda, where lambda in meters.

   use numbatmod
   use class_TriangleIntegrators

   double precision k_0

   integer(8) n_modes, n_msh_el, n_msh_pts
   integer(8) elnd_to_mesh(P2_NODES_PER_EL,n_msh_el)
   double precision v_nd_xy(2,n_msh_pts)
   complex(8) soln_em_e(3,P2_NODES_PER_EL+7,n_modes,n_msh_el)
   complex(8) beta1, t_power
   complex(8) v_beta(n_modes)
   complex(8), dimension(n_modes) :: m_power

   integer(8), intent(out) :: errco
   character(len=EMSG_LENGTH), intent(out) ::  emsg


   ! Locals

   double precision nds_xy(2,P2_NODES_PER_EL)

   complex(8) E_field_el(3,P2_NODES_PER_EL), H_field_el(3,P2_NODES_PER_EL)
   complex(8) Ez_field_el_P3(P3_NODES_PER_EL)

   !  P3 Ez-field
   double precision m_int_p2_p2(P2_NODES_PER_EL, P2_NODES_PER_EL)
   integer(8) j
   integer(8) i_el, ival
   integer(8) nd_i, nd_j, ui
   complex(8) vec_Es(3), vec_H(3)
   complex(8) t_Pz

   type(AnalyticIntegrator) integrator
   type(PyFrontEnd) frontend
   integer(8) ilo, ihi
!
!f2py intent(in) k_0, n_modes, n_msh_el, n_msh_pts
!f2py intent(in) P2_NODES_PER_EL, elnd_to_mesh
!f2py intent(in) x, v_beta, soln_em_e
!
!f2py depend(elnd_to_mesh) P2_NODES_PER_EL, n_msh_el
!f2py depend(x) n_msh_pts
!f2py depend(v_beta) n_modes
!f2py depend(soln_em_e) P2_NODES_PER_EL, n_modes, n_msh_el
!
```



```
!f2py intent(out) m_power


   ui = stdout


   call frontend%init_from_py(n_msh_el, n_msh_pts, elnd_to_mesh, v_nd_xy, errco,
 emsg)
   RETONERROR(errco)

   do ival=1,n_modes
      t_power = D_ZERO
      beta1 = v_beta(ival)

      do i_el=1,n_msh_el

         call frontend%nodes_at_el(i_el, nds_xy)

         call integrator%build_transforms_at(nds_xy, errco, emsg)
         RETONERROR(errco)

!  The matrix m_int_p2_p2 contains the overlap integrals between the P2-polynomi
al basis functions
         call find_overlaps_p2_p2(m_int_p2_p2, integrator%det)

! Need the Et and Ht fields at the P2 nodes
! Getting the Ht fields requires the Ez field which requires the P3 longitudinal
 solutions

!  The components (E_x,E_y) of the mode ival
!  The component E_z of the mode ival.
! The FEM code uses the scaling: E_z = C_IM_ONE* beta1 * \hat{E}_z
         E_field_el = soln_em_e(:, 1:P2_NODES_PER_EL, ival, i_el)

!  E_z-field: The longitudinal component at the P2 vertices, which are also P3 e
lements
         j=3
         Ez_field_el_P3(1:3) = soln_em_e(j, 1:3, ival, i_el)

         j=3
         !  The longitudinal component at the edge nodes and interior node (P3 e
lements)
         ilo = P2_NODES_PER_EL+1
         ihi = P2_NODES_PER_EL+P3_NODES_PER_EL-3
         Ez_field_el_P3(4:P3_NODES_PER_EL) = soln_em_e(j, ilo:ihi, ival, i_el)

         call get_H_field_p3 (k_0, beta1, integrator%mat_T, E_field_el, Ez_field
_el_P3, H_field_el)


         do nd_i=1,P2_NODES_PER_EL
            vec_Es = E_field_el(:, nd_i)

            do nd_j=1,P2_NODES_PER_EL
               vec_H = H_field_el(:, nd_j)

               !   Cross-product Z.(E^* X H) of E^*=vec_Es and H=vec_H
               !TODO: doesn't seem to be conjugating E field. Doesn't matter sin
ce transverse fields are real
               t_Pz = vec_Es(1) * vec_H(2) - vec_Es(2) * vec_H(1)
               t_power = t_power + t_Pz * m_int_p2_p2(nd_i, nd_j)
            enddo
```

```
         enddo

      enddo

      m_power(ival) = t_power
   enddo

end subroutine em_mode_power_sz_analytic
```