```fortran
! Calculate the energy (not power) m_energy integral of an EM mode with itself u
sing
! numerical quadrature.
!
subroutine em_mode_act_energy_int (n_modes, n_msh_el, n_msh_pts,&
   elnd_to_mesh, v_nd_xy, n_elt_mats, el_material, &
   v_refindexn, soln_em_e, m_energy)
!
   use numbatmod
   use class_TriangleIntegrators

   integer(8) n_modes, n_msh_el, n_msh_pts
   integer(8) elnd_to_mesh(P2_NODES_PER_EL,n_msh_el), n_elt_mats
   integer(8) el_material(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) n_lst(n_elt_mats), v_eps(n_elt_mats)
   complex(8), dimension(n_modes) :: m_energy
   integer(8) errco
   character(len=EMSG_LENGTH) emsg

!     Local variables
   integer(8) nod_el_p(P2_NODES_PER_EL)
   complex(8) bas_ovrlap(P2_NODES_PER_EL)
   integer(8) i, j, j1, typ_e
   integer(8) i_el, ival
   integer(8) nd_i, i_eq
   logical  is_curved
   integer(8) n_curved, debug, ui
   double precision nds_xy(2,P2_NODES_PER_EL)

   complex(8) coeff_1
   complex(8) em_E, em_Estar


   type(QuadIntegrator) quadint
   logical do_P3

   double precision t_quadwt


!fo2py intent(in) n_modes, n_msh_el, n_msh_pts
!fo2py intent(in) P2_NODES_PER_EL, elnd_to_mesh, el_material, n_elt_mats
!fo2py intent(in) x, soln_em_e, n_lst
!
!f2py depend(elnd_to_mesh) P2_NODES_PER_EL, n_msh_el
!f2py depend(x) n_msh_pts
!f2py depend(soln_em_e) P2_NODES_PER_EL, n_modes, n_msh_el
!f2py depend(n_lst) n_elt_mats
!f2py depend(el_material) n_msh_el
!
!fo2py intent(out) m_energy
!

   ui = stdout

   do_P3 = .false.

   !     Calculate permittivity
   v_eps = n_lst**2

   m_energy = D_ZERO
```



```fortran
   n_curved = 0
   do i_el=1,n_msh_el
      typ_e = el_material(i_el)


      do j=1,P2_NODES_PER_EL
         nds_xy(:, j) = v_nd_xy(:,  elnd_to_mesh(j,i_el))
      enddo

      is_curved = log_is_curved_elem_tri (P2_NODES_PER_EL, nds_xy)
      if (is_curved) then
         n_curved = n_curved + 1
      endif


      bas_ovrlap = 0.0d0

      do iq=1,quadint%n_quad

         call quadint%build_transforms_at(iq, nds_xy, is_curved, do_P3, errco, e
msg)
         RETONERROR(errco)

         ! transformed weighting of this quadrature point including triangle are
a transform
         t_quadwt = quadint%wt_quad(iq) * abs(quadint%det)

         ! Calculate m_energy of basis functions at quadrature point,
         ! which is a superposition of P2 polynomials for each function (fi_eld)
.
         do nd_i=1,P2_NODES_PER_EL
            bas_ovrlap(nd_i) = bas_ovrlap(nd_i) + &
               t_quadwt * quadint%phi_P2_ref(nd_i) * quadint%phi_P2_ref(nd_i)
         enddo
      enddo

      ! Having calculated m_energy of basis functions on element
      ! now multiply by specific fi_eld values for modes of interest.
      do ival=1,n_modes
         do nd_i=1,P2_NODES_PER_EL
            do i_eq=1,3
               em_Estar = conjg(soln_em_e(i_eq,nd_i,ival,i_el))
               em_E = soln_em_e(i_eq,nd_i,ival,i_el)
               m_energy(ival) = m_energy(ival) + &
                  v_eps(typ_e) * em_Estar * em_E * bas_ovrlap(nd_i)
            enddo
         enddo
      enddo


   enddo

   m_energy = 2.0 * m_energy * SI_EPS_0

end subroutine EM_mode_E_energy_int
```