```fortran
elseif (seec == 4) then ! New model as of Jul12 2013. FEST3Dish?

  Emax = sey%Emax     ! Emax(delta=max,theta=0) in eV
  deltamax = sey%deltamax      ! Maximum secondary electron yield (at Emax) for normal incidence (theta=0)
  E_0 = sey%E_0
  E_0p = sey%E_0p

  if (penf1 <= E_0*4) then      ! Reflect electron

      ! Reflect electron inelastically
!      pvf = reflect_electron(pcol,penf1*(-e),pvf1,2)

      allocate (thetas(1))
      allocate (phis(1))
      allocate (penf(1))
      allocate (vs(1))

      penf = taus88() * penf1
      vs = sqrt(2*penf*(-eme))

      ! Determine which is the normal component of the outgoing velocity
      if (pcol(1)==1 .OR. pcol(3)==1) then
        i = 1              ! Normal component
        j = 2              ! Parallel component(not z)
      elseif (pcol(2)==1 .OR. pcol(4)==1) then
        i = 2              ! Normal component
        j = 1              ! Parallel component (not z)
      endif

      rt = taus88()
      phis = rt * 2 * pi        ! Calculate emission azimuthal angle of secondaries

      rt = taus88()
      thetas = asin(2*rt - 1)   ! Calculate emission angle of secondaries with respect to the normal

      pvf(1,i) = vs(1) * cos(thetas(1))   ! Normal component of secondary velocity

      pvf(1,j) = sin(phis(1)) * vs(1) * sin(thetas(1))
      pvf(1,3) = cos(phis(1)) * vs(1) * sin(thetas(1))

      if (pcol(1) == 1 .OR. pcol(2) == 1) then
        pvf(1,i) = - abs(pvf(1,i))   ! Normal velocity pointing into the inside of the waveguide
      elseif (pcol(3) == 1 .OR. pcol(4) == 1) then
        pvf(1,i) = abs(pvf(1,i))   ! Normal velocity pointing into the inside of the waveguide
      endif

      deallocate (thetas)
      deallocate (phis)
      deallocate (penf)
      deallocate (vs)

  else          ! True secondary(ies)

    ! First calculate yield according to modified Vaughan

    ! Unpack user inputs regarding particle-wall interaction and SEE
    kse = sey%kse           ! Roughness factor for energy, =0 for rough, =1 for dull, =2
```

```fortran
for smooth and anything in between
ks = sey%ks            ! Roughness factor for angle, =0 for rough, =1 for dull, =2 for smooth and anything in between

! Determine secondary electron yields (Vaughan's theory (1993))
Emax = Emax * (1 + kse*theta*theta/(2*pi))
deltamax = deltamax * (1 + ks*theta*theta/(2*pi))

xi = real((penf1 - E_0))/(Emax - E_0)

if (xi <= 3.6) then

  if (xi <= 1) then
    ke = 0.56
  elseif (xi > 1 .OR. xi <= 3.6) then
    ke = 0.25
  endif

    delta = deltamax * ( (xi * exp(1-xi))**ke )

elseif (xi > 3.6) then

    delta = deltamax * 1.125 / (xi**0.35)

endif

if (delta > 10) then
  write(*,*) 'high delta'
  call exit
endif

if (isNaN(real(delta,8))) then
  write(*,*) 'theta = ',theta
  write(*,*) 'penf1 = ',penf1
  write(*,*) 'E_0 = ',E_0
  write(*,*) 'E_0p = ',E_0p
  write(*,*) 'delta = ',delta
  write(*,*) 'E_0p = ',E_0p
  write(*,*) 'E_0 = ',E_0
endif

s = poisdev(real(delta,8),atype)

if (s > 0) then

  allocate (pvf(s,3))
  allocate (yn(s))
  allocate (thetas(s))
  allocate (phis(s))
  allocate (penf(s))
  allocate (vs(s))

  ! Initialize variables
  pvf = 0.        ! m x 3 array of 3D velocities of m secondary electrons
  yn = 0.         ! magnitude of velocity of secondary electrons
  thetas = 0.      ! Emission angle of secondaries with respect to the normal
  phis = 0.        ! Azimuthal emission angle of secondaries
  penf = 0.
  vs = 0.

```

```
      ! Calculate cummulative energy of secondary electrons from a Gaussian distribution
!       Y = -1.
!       do while (Y < 0.)
!         Y = penf1 - Eom * abs(random_normal())
!       enddo
!       Y = sqrt( Y/Eom )    ! v^tilde

      call gratio(real(p_n*s,8), real(penf1/Eom,8), ans, qans, 0)
      rt = taus88()
      ans = ans*rt
      call gaminv(real(p_n*s,8), y, real(0,8), ans, 1-ans, ierr)

      Y = sqrt( Y )    ! v^tilde

      !! Scalar parameters needed to obtain velocity components
      sint = 1.
      do k = 1,s-1
        lnbeta = betaln(real(p_n*(s-k),4),real(p_n,4))

        rt = taus88()
        inbeta = betain( real(rt,8), real(p_n*(s-k),8), real(p_n,8), lnbeta, ifault )
        if (ifault /= 0) then
          write(*,*) 'error calculating incomplete beta function. Terminating.'
          call exit
        endif

        alpha = xinbta( real(p_n*(s-k),8), real(p_n,8), lnbeta, inbeta, ifault )
        if (ifault /= 0) then
          write(*,*) 'error calculating inverse incomplete beta function. Terminating.'
          call exit
        endif

        alpha = asin( sqrt( alpha ) )   ! Alpha angles used to calculate the magnitude of velocity

        yn(k) = Y * sint * cos(alpha)   ! magnitude of outgoing velocities

        sint = sint * sin(alpha)                ! Spherical coordinates factor
      enddo
      yn(s) = Y * sint

      ! Determine which is the normal component of the outgoing velocity
      if (pcol(1)==1 .OR. pcol(3)==1) then
        i = 1              ! Normal component
        j = 2              ! Parallel component(not z)
      elseif (pcol(2)==1 .OR. pcol(4)==1) then
        i = 2              ! Normal component
        j = 1              ! Parallel component (not z)
      endif

      do k = 1,s
        penf(k) = Eom * (yn(k))**2
        vs(k) = sqrt(2*penf(k)*(-eme))

        rt = taus88()
        phis(k) = rt * 2 * pi        ! Calculate emission azimuthal angle of secondaries

        rt = taus88()
        thetas(k) = asin(2*rt - 1)   ! Calculate emission angle of secondaries with respect to the normal

```

```
        pvf(k,i) = vs(k) * cos(thetas(k))   ! Normal component of secondary velocity
        pvf(k,j) = sin(phis(k)) * vs(k) * sin(thetas(k))
        pvf(k,3) = cos(phis(k)) * vs(k) * sin(thetas(k))
      enddo

!       write(*,*) 's = ',s
!       write(*,*) 'penf1 = ', penf1
!       write(*,*) 'penf(:) = ', penf
!       write(*,*) 'vs(:) = ', vs
!       stop

      if (pcol(1) == 1 .OR. pcol(2) == 1) then
        do k = 1,s
          pvf(k,i) = - abs(pvf(k,i))   ! Normal velocity pointing into the inside of the waveguide
        enddo
      elseif (pcol(3) == 1 .OR. pcol(4) == 1) then
        do k = 1,s
          pvf(k,i) = abs(pvf(k,i))   ! Normal velocity pointing into the inside of the waveguide
        enddo
      endif

      deallocate (yn)
      deallocate (thetas)
      deallocate (phis)
      deallocate (penf)
      deallocate (vs)

    elseif (s == 0) then
      allocate (pvf(1,3))
      pvf = 0.
    endif

  endif

endif
```