## Stratified Resampling

```
!=======================================================================
  subroutine PF_StraResamp(wght_v,keep_v)
!-----------------------------------------------------------------------
! Purpose: resamples the weights of the particles
! Method:  stratified resampling
! Author:
! 02 oct 2009 : Douglas Plaza G.
!-----------------------------------------------------------------------
  use precision
  use clm_varder
  use clm_varctl    , only : print_stat_out
  use clm_varmap    , only : begpatch, endpatch, numland
  use clm_varpar    , only : maxpatch_pft
  implicit none
!----Arguments----------------------------------------------------------
  real(r8), dimension(:), intent(inout)  :: keep_v           !vector with particle survivals
  real(r8), dimension(:), intent(inout)  :: wght_v           !particle state vector
!----Local Variables----------------------------------------------------
  real(r8), dimension(maxpatch_pft)     :: di,ind1,sel, W_cs
  integer,  dimension(maxpatch_pft)     :: tmp_kpt
  real(r8)                              :: kon
  integer                               :: ctr, i, k, r, l
!----End Variable List--------------------------------------------------
!--- variables initialization ------
  kon     = 0.0
  di(:)   = 0.0
  ind1(:) = 0.0
  sel(:)  = 0.0
  W_cs(:) = 0.0
!-----------------------------------
  di(1) = 1./(2*maxpatch_pft)!kon/2.
  do i = 2,maxpatch_pft    !+1 <== bus error (dimensions)
    di(i) = di(i-1)+ (1./maxpatch_pft)!kon
  enddo
  kon     =1./maxpatch_pft
  call random_seed
  call random_number(ind1)
  sel     = di + (ind1*kon - kon/2.)
  W_cs(1) = wght_v(1)

  do i=2,maxpatch_pft
    W_cs(i) = W_cs(i-1)+wght_v(i)
  enddo
  ctr = 1
  do i = 1,maxpatch_pft
    do while ((ctr <= maxpatch_pft).and.(sel(ctr) < W_cs(i)))
      tmp_kpt(ctr) = i
      ctr          = ctr+1
    enddo
  enddo
  keep_v = tmp_kpt
  end subroutine PF_StraResamp
!=======================================================================
```

## Residual resampling

```
!======================================================================
  subroutine PF_ResResamp(wght_v,keep_v)
!----------------------------------------------------------------------
! Purpose: resamples the weights of the particles
! Method:  residual resampling
! Author:
! 15 ago 2011 : Douglas Plaza G.
!----------------------------------------------------------------------
  use precision
  use clm_varder
  use clm_varctl    , only : print_stat_out
  use clm_varmap    , only : begpatch, endpatch, numland
  use clm_varpar    , only : maxpatch_pft
  implicit none
!----Arguments---------------------------------------------------------
  real(r8), dimension(:), intent(inout)  :: keep_v   !vector with particle survivals
  real(r8), dimension(:), intent(inout)  :: wght_v   !importance weights
!----Local Variables---------------------------------------------------
  real(r8), dimension(maxpatch_pft)     :: W_res, cumD
  integer,  dimension(maxpatch_pft)     :: N_child, inInd, outInd
  integer                               :: N_res, ii,inde,jj,c_flag,i
!---Allocatable variables----------------------------------------------
  real(r8), dimension(:), allocatable   :: N_rand, rand_out, u_out, temp_Uout
  integer, dimension(:),allocatable     :: N_res_v
!----End Variable List-------------------------------------------------
!--- variables initialization ------
  N_child(:) = 0
  outInd(:)  = 0
  W_res(:)   = 0.0
  cumD(:)    = 0.0
!-----------------------------------
  do i = 1,maxpatch_pft
    W_res(i)    =  maxpatch_pft*wght_v(i)
    N_child(i)  =  int(W_res(i))
  enddo
  N_res = maxpatch_pft - sum(N_child)
  allocate (N_rand(N_res), rand_out(N_res), u_out(N_res), temp_Uout(N_res))
  allocate (N_res_v(N_res))
  c_flag = 1
  do i = N_res,1,-1
    N_res_v(c_flag) = i
    c_flag = c_flag+1
  enddo
  if (N_res .ne. 0) then
    do i = 1,maxpatch_pft
      W_res(i) = (W_res(i)-N_child(i))/N_res
      if (i .eq. 1) then
        cumD(i) = W_res(1)
      else
        cumD(i) = cumD(i-1)+W_res(i)
      endif
    enddo
    call random_seed
    call random_number(N_rand)
    do i=1,N_res
```

```
    rand_out(i) = N_rand(i)**(1./N_res_v(i))
    if (i .eq. 1) then
      u_out(i) = rand_out(1)
    else
      u_out(i) = u_out(i-1)*rand_out(i)
    endif
  enddo
  temp_Uout = u_out
  ii = N_res
  do i=1,N_res
    u_out(i) = temp_Uout(ii)
    ii=ii-1
  enddo
  jj = 1
  do i=1,N_res
    do while (u_out(i) .gt. cumD(jj))
      jj=jj+1
    enddo
    N_child(jj)=N_child(jj)+1
  enddo
  write(*,*)"The value of N_child is: "
  write(*,*)(N_child(i),i=1,maxpatch_pft)
 endif
 do i=1,maxpatch_pft
   inInd(i) = i
 enddo
 inde = 1;
 do i=1,maxpatch_pft
  if (N_child(i) .gt. 0) then
    do jj=inde,inde+N_child(i)-1
      outInd(jj) = inInd(i)
    enddo
  endif
  inde = inde+N_child(i)
 enddo
 write(*,*)"The value of outInd is: "
 write(*,*)(outInd(i),i=1,maxpatch_pft)
 keep_v = outInd
 deallocate (N_rand, rand_out, u_out, temp_Uout)
 deallocate (N_res_v)

 end subroutine PF_ResResamp
!=======================================================================
```