Supplementary material

1. Parameters

The parameters that are defined outside the subroutines presented in this document are shown here.

```
R_earth              = 6.371221E6_dp                    ! Radius of the Earth in meters
dx_anice40           = 40000._dp                        ! Grid scale of ANICE: 40km for Nam, Eas, Ant; 20km for Grl
ice_density          =  910.0_dp                        ! Ice density in kg m^-3
seawater_density     = 1028.0_dp                        ! Seawater density in kg m^-3

C_SLE%NP             = 2 * RES * (RES-1) * 20 + 12      ! All SELEN elements = 141612 (with RES = 60), equation (A1)
C_SLE%NEL_ANICE      = 26012                            ! All ice sheet elements of SELEN
C_SLE%NN_NAM         = NX_nam * NY_nam                  ! Number of grid points of North America  = 21901
C_SLE%NN_EAS         = NX_eas * NY_eas                  ! Number of grid points of Eurasia        = 17955
C_SLE%NN_GRL         = NX_grl * NY_grl                  ! Number of grid points of Greenland      = 10857
C_SLE%NN_ANT         = NX_ant * NY_ant                  ! Number of grid points of Antarctica     = 19881
C_SLE%ALFA           = SQRT(4._dp / DBLE(C_SLE%NP))     ! Radius of SELEN elements, equation (A3)
C_SLE%radice40       = dx_anice40 / (SQRT(C%pi) * R_earth ! Radius of ANICE grid boxes, equation (A4) (20km for GRL)
```

2. Initial files for SELEN

This subroutine is used to include the ANICE ice loading and topography in the global fields of SELEN.
One example is given for the North American ice sheet, subroutines for the other four ice sheets are practically the same, see the different IF-statements on line 177-191. For Greenland an additional masking is used, to only include land points (lines 202,203 and lines 222,223). Moreover, the grid scale for North America, Eurasia and Antarctica is 40 km, for Greenland this is 20 km.

```
MODULE create_initial_selen_module
! Program to create the initial files for selen, first do this then run the Coupled model
!
! The order of the ice sheets is NAM, EAS, GRL, ANT
!
  USE grid_main_module, ONLY: dp, C                        ! Main definitions of parameters

  ! Longitude, latitude, initial ice thickness loading and initial bedrock topography of the four ice sheet in ANICE
  ! GRL_mask_ref is the land mask of Greenland (including 1 grid point of ocean around),
  ! used to determine the elements for Greenland in SELEN (see Fig. 3b)
  USE grid_selen_module, ONLY: C_SLE, NAM_Long, NAM_Lat, NAM_Load_ref, NAM_Topo_ref,               &
                                      EAS_Long, EAS_Lat, EAS_Load_ref, EAS_Topo_ref,               &
                                      GRL_Long, GRL_Lat, GRL_Load_ref, GRL_Topo_ref, GRL_mask_ref, &
                                      ANT_Long, ANT_Lat, ANT_Load_ref, ANT_Topo_ref
  USE parameters_main_module, ONLY: R_earth, ice_density, seawater_density

  IMPLICIT NONE

  INTEGER :: LOCAL_NEL

CONTAINS
  SUBROUTINE anice2pixels()
    integer  :: n_ice
    integer  :: i

    INTEGER,  DIMENSION(C_SLE%NP) :: anclonp, anclatp  ! anchor points of each element
    INTEGER,  DIMENSION(C_SLE%NP) :: of, fgi           ! ocean function (1=topo below sea level)
    INTEGER,  DIMENSION(C_SLE%NP) :: of, fgi           ! floating function (1=floating ice, so no loading)
    INTEGER,  DIMENSION(C_SLE%NP) :: icelab            ! label for the ice sheets, 1=nam, 2=eas, 3=grl, 4=ant

    REAL(dp), DIMENSION(C_SLE%NP) :: lonp, latp        ! longitude and latitude for all element
    REAL(dp), DIMENSION(C_SLE%NP) :: hiel, topo        ! interpolated ice thickness, topography for all elements
!
! -- Open initial file of all SELEN elements
!
 open(1,file='px-table-topo-res60-etopo1.dat',status='unknown')  ! new file: px-table-topo-res60-etopo1.dat
 do i=1, C_SLE%NP
  read(1,*) lonp(i), latp(i), anclonp(i), anclatp(i), topo(i), of(i)
 end do
 close(1)

icelab = 0
hiel   = 0._dp
```

```
!
! -- Loop over the 4 ice sheets: Nam, Eas, Grl, Ant
!
  call trig_converter_nam(C_SLE%NN_NAM,lonp,latp,anclonp,anclatp,topo,of,hiel,icelab)
  call trig_converter_eas(C_SLE%NN_EAS,lonp,latp,anclonp,anclatp,topo,of,hiel,icelab)
  call trig_converter_grl(C_SLE%NN_GRL,lonp,latp,anclonp,anclatp,topo,of,hiel,icelab)
  call trig_converter_ant(C_SLE%NN_ANT,lonp,latp,anclonp,anclatp,topo,of,hiel,icelab)

  ! determine floating ice and set loading to zero
  where (-topo*seawater_density  >  hiel*ice_density) fgi = 1
  where (-topo*seawater_density  <= hiel*ice_density) fgi = 0
  where (fgi==1) hiel = 0._dp
!
! -- Write new file for all elements new ice loading and ocean function corrected for floating ice
!
 open(100,file='selen-px-topo-res60.pixelized',status='unknown',recl=4000)
 do i=1,C_SLE%NP
  write(100,*) lonp(i), latp(i), anclonp(i), anclatp(i), of(i) * fgi(i), topo(i), hiel(i)
 end do
 close(100)
!
! -- Write new file with specific ANICE elements within SELEN
!
 open(200,file='ice_loaded_pixels_anice_res60',status='unknown',recl=4000)
!
 n_ice=0
 do i=1, C_SLE%NP
   ! Here icelab is saved and used in the model as ANICE_sle_icelab to define the SELEN elements for each ice sheet
   if(icelab(i) /= 0) then
     n_ice=n_ice+1
     write(200,*) lonp(i), latp(i), anclonp(i), anclatp(i), icelab(i)
   end if
 end do

 LOCAL_NEL = n_ice   ! which is defined as NEL_ANICE = 26012

  END SUBROUTINE anice2pixels
!
! -- Example for North America
!
  SUBROUTINE trig_converter_nam(nice,lonp,latp,anclonp,anclatp,topo,of,hiel,icelab)

    INTEGER,                        INTENT(IN)    :: nice
    INTEGER,  DIMENSION(C_SLE%NP), INTENT(IN)    :: anclonp, anclatp
    REAL(dp), DIMENSION(C_SLE%NP), INTENT(IN)    :: lonp, latp

    ! These fields are adjusted with the ANICE values
    INTEGER,  DIMENSION(C_SLE%NP), INTENT(INOUT) :: of           ! ocean function for global SELEN
    REAL(dp), DIMENSION(C_SLE%NP), INTENT(INOUT) :: topo         ! topography for global SELEN

    INTEGER,  DIMENSION(C_SLE%NP), INTENT(OUT)   :: icelab       ! ice labels for global SELEN
    REAL(dp), DIMENSION(C_SLE%NP), INTENT(OUT)   :: hiel         ! ice thickness for global SELEN

    INTEGER                                      :: i, j
    INTEGER                                      :: pixcount,icecount

    REAL(dp)                                     :: intarea, deg
    REAL(dp)                                     :: volice, voltopo
    REAL(dp)                                     :: hp, bp
    REAL(dp)                                     :: volratio, vol, volkm

    INTEGER,  dimension(:), allocatable          :: anclontemp, anclatemp
    REAL(dp), dimension(:), allocatable          :: lontemp, latemp, htemp, btemp
    REAL(dp), dimension(:), allocatable          :: hi
!
! -- Allocate arrays
!
 allocate (hi         (nice      ))
 allocate (lontemp    (1:C_SLE%NP))
 allocate (latemp     (1:C_SLE%NP))
 allocate (anclontemp (1:C_SLE%NP))
 allocate (anclatemp  (1:C_SLE%NP))
 allocate (htemp      (1:C_SLE%NP))
 allocate (btemp      (1:C_SLE%NP))
!
! -- calculate volume of ice of rectangular grid of ANICE
!
    volkm = 0._dp
    do i=1, nice
      volkm = volkm + C_SLE%dx_anice40 * C_SLE%dx_anice40 * NAM_Load_ref(i)
    end do
!
! -- calculate volume of ice for circular grid points of ANICE
!
    vol = 0._dp
    do i=1, nice
      vol = vol + 2._dp * C%pi * R_earth**2 * NAM_Load_ref(i) * (1._dp - COS(C_SLE%radice40))
    end do
```

```
 if (vol == 0._dp .and. volkm == 0._dp) then
   volratio = 1._dp
 else if (vol > 0. .and. volkm > 0.) then
   volratio = vol / volkm
 end if
!
! -- set the ice thickness from the original ice-sheet grid
!
 do i=1,nice
  hi(i) = NAM_Load_ref(i)
 end do
!
! -- calculate volume of ice on radial elements
!
    vol = 0._dp
    do i=1, nice
      vol = vol + 2._dp * C%pi * R_earth**2 * hi(i) * (1._dp - COS(C_SLE%radice40))
    end do
!
! -- interpolate the ice thickness to the SELEN elements
!
pixcount=0

! Loop over all ANICE elements of SELEN
DO i=1, C_SLE%NP

!
! -- within each subroutine of the four regions, specific boundaries of the SELEN elements
!    are set to circumvent SELEN elements that do not cover that specific region
!
    ! Specific boundaries for North America (in trig_converter_nam)
    IF (latp(i) >= 30._dp .and. latp(i) <= 85._dp .and. lonp(i) >= 180._dp .and. lonp(i) <= 350._dp) THEN

    ! Specific boundaries for Eurasia (in trig_converter_eas)
    IF (latp(i) >= 40._dp .and. (lonp(i) >= 305._dp .or. lonp(i) <= 135._dp) ) THEN

    ! Specific boundaries for Greenland (in trig_converter_grl)
    IF (latp(i) >= 54._dp .and. latp(i) <= 90._dp .and. (lonp(i) >= 165._dp .or. lonp(i) <= 15._dp) ) THEN

    ! Specific boundaries for Antarctica (in trig_converter_ant)
    IF(latp(i) <= -50._dp) THEN

      hp = 0._dp
      bp = 0._dp

      icecount = 0
      volice   = 0._dp
      voltopo  = 0._dp

      ! loop over the ice elements (converted into disks!)
      DO j=1, nice
        ! only in trig_converter_grl this is included for the Greenland ANICE points
        ! IF (GRL_mask_ref(j) == 1) THEN

        ! determine the angular distance between the SELEN and ANICE location
        deg = angdist(lonp(i),latp(i),NAM_Long(j),NAM_Lat(j))

        ! calculate the overlapping area of the two spheres
        if ( C%rad2deg * min(C_SLE%ALFA, C_SLE%radice40) <= (C%rad2deg * max(C_SLE%ALFA,C_SLE%radice40) - deg) ) then
          intarea  = 2._dp * C%pi * R_earth**2 * (1._dp - COS(MIN(C_SLE%ALFA,C_SLE%radice40)))
          icecount = icecount + 1
          volice   = volice   + intarea * hi(j)
          voltopo  = voltopo  + intarea * NAM_Topo_ref(j)

        else if ( C%rad2deg * (C_SLE%ALFA + C_SLE%radice40) > deg ) then
          intarea  = optintersect(C_SLE%ALFA, C_SLE%radice40, deg)
          icecount = icecount + 1
          volice   = volice   + intarea * hi(j)
          voltopo  = voltopo  + intarea * NAM_Topo_ref(j)
        end if

       ! only in trig_converter_grl this is included for the Greenland ANICE points
       ! END IF
      END DO

      ! set ice thickness on the SELEN elements and set ice label
      if (icecount > 0) then
        icelab(i) = 1    ! 1 for Nam, 2 for Eas, 3 for Grl, 4 for Ant
        pixcount  = pixcount + 1

        lontemp(pixcount) = lonp(i)
        latemp(pixcount)  = latp(i)

        anclontemp(pixcount) = anclonp(i)
        anclatemp(pixcount)  = anclatp(i)
```

```
          hp = volice  / (2._dp * C%pi * R_earth**2 * (1._dp - COS(C_SLE%ALFA))) ! = area of pixel
          bp = voltopo / (2._dp * C%pi * R_earth**2 * (1._dp - COS(C_SLE%ALFA))) ! = area of pixel

          htemp(pixcount) = hp
          btemp(pixcount) = bp
        end if

      end if
    END DO ! end of loop over all elements NP
!
! -- calculate volume of ice:
!
    vol = 0._dp
    do i=1, pixcount
      vol = vol + 2._dp * C%pi * R_earth**2 * htemp(i) * (1._dp - COS(C_SLE%ALFA))
    end do
!
! -- calculate the new volume ratio and correct the ice load
!
    if (vol == 0._dp .and. volkm == 0._dp) then
      volratio = 1._dp
    else if (vol > 0._dp .and. volkm > 0._dp) then
      volratio = vol / volkm
    endif

    do i=1,pixcount
      htemp(i) = htemp(i) / volratio
    end do
!
! -- set new ice load and topography in the global SELEN arrays
!
    do i=1,C_SLE%NP
      do j=1, pixcount
        if(anclonp(i) == anclontemp(j) .and. anclatp(i) == anclatemp(j)) then
          hiel(i) = htemp(j)
          topo(i) = btemp(j)
          if(topo(i) >  0) of(i) = 0
          if(topo(i) <= 0) of(i) = 1
        end if
      end do
    end do

    deallocate (hi)
    deallocate (lontemp)
    deallocate (latemp)
    deallocate (anclontemp)
    deallocate (anclatemp)
    deallocate (htemp)

  END SUBROUTINE trig_converter_nam
```

3. Interpolation of ice loading

This subroutine is used to interpolate the ice loading of ANICE on the SELEN grid. The basics are quite similar to the subroutine shown in section 2, only here the interpolation is performed for ice loading only. Again, we provide one example for North America, the routine is the same for the other models. Similarly, the grid scale for North America, Eurasia and Antarctica is 40 km, for Greenland this is 20 km.

```
MODULE interpolation_selen_module
  ! This model is used to interpolate the ice loading from the ANICE grid (e.g. NN_GRL)
  ! to the elements containing ice on the SELEN grid: NEL_ANICE

  USE grid_main_module, ONLY: dp, C                          ! Main definitions of parameters
  USE grid_selen_module, ONLY: C_SLE, NAM_Long, NAM_Lat, &   ! Longitude and latitude of the North American ice sheet
                                  EAS_Long, EAS_Lat, &       ! Longitude and latitude of the Eurasian ice sheet
                                  GRL_Long, GRL_Lat, &       ! Longitude and latitude of the Greenland ice sheet
                                  ANT_Long, ANT_Lat          ! Longitude and latitude of the Antarctic ice sheet

  USE reference_selen_module, ONLY: ANICE_sle_lon, &         ! Longitude of SELEN elements
                                 ANICE_sle_lat, &            ! Latitudes of SELEN elements
                                 ANICE_sle_icelab            ! Ice label of SELEN elements, specific for each
                                                             !  of the four ice sheets: 1=Nam, 2=Eus, 3=Grl, 4=Ant
  USE parameters_main_module, ONLY: R_earth                  ! Radius of the Earth

  IMPLICIT NONE

CONTAINS
  SUBROUTINE interpolate_ice(nam_load, eas_load, grl_load, ant_load, int_ice_anice)
    ! Interpolating ice loading of the ANICE grid to the ice elements of SELEN

    ! Input:
    REAL(dp), DIMENSION(C_SLE%NN_NAM),    INTENT(IN)  :: nam_load       ! Ice thickness on land from North America
    REAL(dp), DIMENSION(C_SLE%NN_EAS),    INTENT(IN)  :: eas_load       ! Ice thickness on land from Eurasia
    REAL(dp), DIMENSION(C_SLE%NN_GRL),    INTENT(IN)  :: grl_load       ! Ice thickness on land from Greenland
    REAL(dp), DIMENSION(C_SLE%NN_ANT),    INTENT(IN)  :: ant_load       ! Ice thickness on land from Antarctica

    ! Output variables:
    REAL(dp), DIMENSION(C_SLE%NEL_ANICE), INTENT(OUT) :: int_ice_anice  ! The interpolated ice thickness


    WRITE(0,*) 'running interpolate_ice'

    ! Initialize the interpolated data is equal to the initial ice thickness:
    int_ice_anice = 0._dp

    ! Call the interpolation routines for each ice sheet, depending on which ice sheets are used
    ! Determine which ice sheets are used in the model: Nam, Eas, Grl, Ant
    if(C%which_icesheets(1:1) == 'T') call trig_converter_nam(nam_load, int_ice_anice)
    if(C%which_icesheets(2:2) == 'T') call trig_converter_eas(eas_load, int_ice_anice)
    if(C%which_icesheets(3:3) == 'T') call trig_converter_grl(grl_load, int_ice_anice)
    if(C%which_icesheets(4:4) == 'T') call trig_converter_ant(ant_load, int_ice_anice)

  END SUBROUTINE interpolate_ice
!
! North America --- Only 1 example is show here, the subroutines for the other ice sheets are identical
!
  SUBROUTINE trig_converter_nam(nam_load, int_load_nam)
!
    ! Input: ice thickness on land from North America of ANICE
    REAL(dp), DIMENSION(C_SLE%NN_NAM),    INTENT(IN)    :: nam_load

    ! Output: interpolated ice thickness on SELEN elements
    REAL(dp), DIMENSION(C_SLE%NEL_ANICE), INTENT(INOUT) :: int_load_nam

    INTEGER                                             :: i,j

    REAL(dp)                                            :: intarea, deg
    REAL(dp)                                            :: volice, vol, volkm, volratio
    REAL(dp)                                            :: hp
    INTEGER                                             :: icecount, pixcount

    REAL(dp), dimension(C_SLE%NEL_ANICE)                :: htemp
    REAL(dp), dimension(C_SLE%NN_NAM)                   :: hi

!
! -- calculate volume of ice of rectangular grid of the ice-sheet model ANICE
!
    volkm = 0._dp
    do i=1, C_SLE%NN_NAM
      volkm = volkm + C_SLE%dx_anice40 * C_SLE%dx_anice40 * nam_load(i)  ! grid scale of Nam is 40 km
    end do
```

```
!
! -- calculate volume of ice for circular grid points of ANICE
!
   vol = 0._dp
   do i=1, C_SLE%NN_NAM
     vol = vol + 2._dp * C%pi * R_earth**2 * nam_load(i) * (1._dp - COS(C_SLE%radice40))
   end do
!
! -- calculate the ratio of the volume from the circular grid with that from the original ice grid
!
   if (vol == 0._dp .and. volkm == 0._dp) then
     volratio = 1._dp
   else if (vol > 0._dp .and. volkm > 0._dp) then
     volratio = vol / volkm
   end if
!
! -- set the ice thickness from the original ice-sheet grid
!
   do i=1,C_SLE%NN_NAM
     hi(i) = nam_load(i)
   end do
!
! -- interpolate the ice thickness to the SELEN elements
!
  pixcount = 0                   ! counting the number of pixels for the ice sheet
!
! --  Loop over all ANICE elements
!
  DO i=1, C_SLE%NEL_ANICE
    ! Specific boundaries for North America
    IF (ANICE_sle_icelab(i) == 1) THEN
      hp       = 0._dp
      icecount = 0
      volice   = 0._dp

      ! loop over the ice elements (converted into disks!)
      DO j=1, C_SLE%NN_NAM
        ! determine the angular distance between the SELEN and ANICE location
        deg = angdist(ANICE_sle_lon(i), ANICE_sle_lat(i), NAM_Long(j), NAM_Lat(j))

        ! calculate the overlapping area of the two spheres
        if ( C%rad2deg * MIN(C_SLE%ALFA, C_SLE%radice40) <= (C%rad2deg * max(C_SLE%ALFA,C_SLE%radice40) - deg) ) then
          intarea  = 2._dp * C%pi * R_earth**2 * (1._dp - COS(MIN(C_SLE%ALFA,C_SLE%radice40)))
          icecount = icecount + 1
          volice   = volice   + intarea * hi(j)
        else if ( C%rad2deg *(C_SLE%ALFA + C_SLE%radice40) > deg ) then
          intarea  = optintersect(C_SLE%ALFA, C_SLE%radice40, deg)
          icecount = icecount + 1
          volice   = volice   + intarea * hi(j)
        end if
      END DO

      ! count the number of pixels
      if (icecount > 0) then
        pixcount  = pixcount + 1

        hp = volice  / (2._dp * C%pi * R_earth**2 * (1._dp - COS(C_SLE%ALFA))) ! = volume divided by the area of pixel
        htemp(i) = hp
      end if

    END IF
  END DO ! end of loop over all elements NEL_ANICE
!
! -- calculate the new volume of ice on the SELEN elements:
!
   vol = 0._dp
   do i=1, C_SLE%NEL_ANICE
     if (ANICE_sle_icelab(i) == 1) then
       vol = vol + 2._dp * C%pi * R_earth**2 * htemp(i) * (1._dp - COS(C_SLE%ALFA))
     end if
   end do
!
! -- calculate the new volume ratio and correct the ice load
!
   if (vol == 0._dp .and. volkm == 0._dp) then
     volratio = 1._dp
   else if (vol > 0._dp .and. volkm > 0._dp) then
     volratio = vol / volkm
   endif
!
! -- correct the interpolated ice thickness on the SELEN elements to conserve ice volume
!
   WHERE(ANICE_sle_icelab == 1) int_load_nam = htemp / volratio
!
  END SUBROUTINE trig_converter_nam
```

```
!
! ========================================================================================
!
  ! Function to calculate the angular distance between two points
  FUNCTION angdist(lonp, latp, loni, lati) RESULT(deg)

    REAL(dp) :: lonp, latp, loni, lati

    ! Results:
    REAL(dp) :: deg

    deg =  C%rad2deg * ACOS( (COS(C%deg2rad * (90._dp - latp)) * COS(C%deg2rad * (90._dp - lati)) ) + &
                             (SIN(C%deg2rad * (90._dp - latp)) * SIN(C%deg2rad * (90._dp - lati))   * &
                              COS(C%deg2rad * (lonp-loni)) ))
  END FUNCTION angdist

!
! ========================================================================================
!
  ! Function to calculate the intersectional area of two spheres
  FUNCTION optintersect(radp, radi, deg) RESULT(intarea)

    REAL(dp) :: radp, radi, deg

    integer, parameter :: a=0
    integer, parameter :: b=1

    REAL(dp)           :: x         ! = (1-(deg-abs(radp-radi))/(radp+radi-abs(radp-radi)))
    REAL(dp)           :: smooth    ! = ( (-2*((x-a/b-a))**3) + (3*((x-a)/(b-a))**2) )
    ! Result
    REAL(dp)           :: intarea

    x       = 1._dp - (deg- C%rad2deg * abs(radp-radi)) / (C%rad2deg * (radp+radi-abs(radp-radi)))
    smooth  = (-2._dp * ( (x-dble(a)/dble(b-a)))**3) + (3._dp*((x-dble(a))/(dble(b-a)))**2)

    intarea = 2._dp * C%pi * R_earth**2 * (1._dp - COS(MIN(radp,radi))) * smooth

  END FUNCTION optintersect
!
!
END MODULE interpolation_selen_module
```