# SICE
in
# SURFEX 8.1

details of implementation and recent developments

Yurii Batrak (MET-Norway)

XVIII·MAR·MMXIX

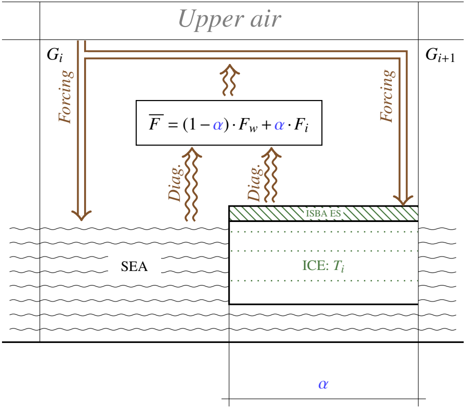
# ALADIN-HIRLAM NWP system cy43h2 uses SURFEX 8.1

- New SURFEX code misses some components that are in use in some operational NWP systems run within the HIRLAM consortium
- One of them is the thermodynamic sea ice scheme SICE

# Default SICE configuration

SICE scheme provides the prognostic ice surface temperature

- ice covered areas are defined by the ice concentration field
- ice thickness is uniform and fixed
- snow-free configuration

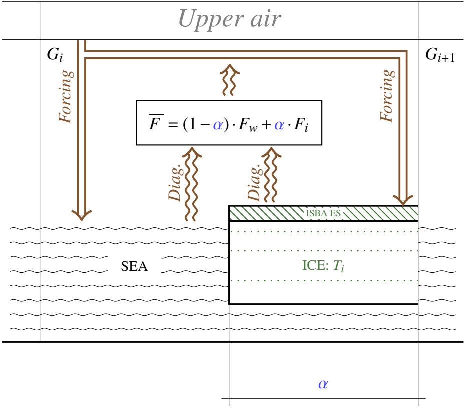


Sea ice as seen by SICE

# Extended SICE configuration

SICE scheme provides the prognostic ice surface temperature, ice thickness and snow on ice state

- ice covered areas are defined by the ice concentration field
- ice thickness is prognostic
- snow on ice is resolved by ISBA-ES



Sea ice as seen by SICE

# How to put SICE in SURFEX 8.1?

- SURFEX 8 provides it own sea ice model GELATO
- Both SICE and GELATO should be available for users
- It is preferable to have two schemes to provide a similar interface
- But schemes should not interfere with each other

# How to put SICE in SURFEX 8.1? Classic approach.

```
! coupling_seafluxn.F90
IF (S%CSEAICE_SCHEME=='GELATO') THEN
  CALL SEAICE_GELATO1D_n(S, HPROGRAM,PTIMEC, PTSTEP)
END IF
! ...
IF (S%LHANDLE_SIC) THEN
  IF (S%CSEAICE_SCHEME/='GELATO') THEN
     S%XTICE    = S%XSST
     S%XSIC     = S%XFSIC
     S%XICE_ALB = XALBSEAICE
  END IF
! ...
```

- Traditionally in SURFEX used string-based selectors to identify the scheme in use
- They allow fine-grained control over the code but have drawbacks

# How to put SICE in SURFEX 8.1? Classic approach.

```
! coupling_seafluxn.F90
IF (S%CSEAICE_SCHEME=='GELATO') THEN
  CALL SEAICE_GELATO1D_n(S, HPROGRAM,PTIMEC, PTSTEP)
END IF
! ...
IF (S%LHANDLE_SIC) THEN
  IF (S%CSEAICE_SCHEME/='GELATO') THEN
     S%XTICE    = S%XSST
     S%XSIC     = S%XFSIC
     S%XICE_ALB = XALBSEAICE
  END IF
! ...
```

- Traditionally in SURFEX used string-based selectors to identify the scheme in use
- They allow fine-grained control over the code but have drawbacks

schemes are not enforced to follow the same interface

# How to put SICE in SURFEX 8.1? Classic approach.

```
! coupling_seafluxn.F90
IF (S%CSEAICE_SCHEME=='GELATO') THEN
  CALL SEAICE_GELATO1D_n(S, HPROGRAM,PTIMEC, PTSTEP)
END IF
! ...
IF (S%LHANDLE_SIC) THEN
  IF (S%CSEAICE_SCHEME/='GELATO') THEN
     S%XTICE    = S%XSST
     S%XSIC     = S%XFSIC
     S%XICE_ALB = XALBSEAICE
  END IF
! ...
```

- Traditionally in SURFEX used string-based selectors to identify the scheme in use
- They allow fine-grained control over the code but have drawbacks

schemes are not enforced to follow the same interface

adding a new scheme requires to update all these **`IF`**-statements

# How to put SICE in SURFEX 8.1? Classic approach.

```
! coupling_seafluxn.F90
IF (S%CSEAICE_SCHEME=='GELATO') THEN
  CALL SEAICE_GELATO1D_n(S, HPROGRAM,PTIMEC, PTSTEP)
END IF
! ...
IF (S%LHANDLE_SIC) THEN
  IF (S%CSEAICE_SCHEME/='GELATO') THEN
     S%XTICE    = S%XSST
     S%XSIC     = S%XFSIC
     S%XICE_ALB = XALBSEAICE
  END IF
! ...
```

- Traditionally in SURFEX used string-based selectors to identify the scheme in use
- They allow fine-grained control over the code but have drawbacks

schemes are not enforced to follow the same interface

adding a new scheme requires to update all these **`IF`**-statements

but if one of the is missed compiler would not complain

# How to put SICE in SURFEX 8.1? New approach.

- Fortran 2003 adds extensive OOP capabilities to the language
- They could be used to enforce the common interface for SICE and GELATO
- The calling side would operate the schemes via this interface
- As result it does not need any knowledge about the ice scheme in use

A drawback of this solution is that the both schemes should fully implement the common interface

# How to put SICE in SURFEX 8.1? New approach.

- Fortran 2003 adds extensive OOP capabilities to the language
- They could be used to enforce the common interface for SICE and GELATO
- The calling side would operate the schemes via this interface
- As result it does not need any knowledge about the ice scheme in use

A drawback of this solution is that the both schemes should fully implement the common interface

Adding a new method requires support from both SICE and GELATO

# How to put SICE in SURFEX 8.1? New approach.

- Fortran 2003 adds extensive OOP capabilities to the language
- They could be used to enforce the common interface for SICE and GELATO
- The calling side would operate the schemes via this interface
- As result it does not need any knowledge about the ice scheme in use

A drawback of this solution is that the both schemes should fully implement the common interface

Adding a new method requires support from both SICE and GELATO

Calling side should not use any scheme-specific variables

# Common interface of a sea ice scheme in SURFEX

```
TYPE, PUBLIC, ABSTRACT :: SEA_ICE_t
    REAL, POINTER :: XSEABATHY(:) !< bathymetry
    REAL, POINTER :: XSST(:) !< sea surface temperature [K]
    REAL, POINTER :: XSSS(:) !< se surface salinity [K]

    LOGICAL :: LINTERPOL_SIC
    LOGICAL :: LINTERPOL_SIT
  CONTAINS
    PROCEDURE(IINIT), DEFERRED, PASS :: INIT
    PROCEDURE(IPREP), DEFERRED, PASS :: PREP
    PROCEDURE(IASSIM), DEFERRED, PASS :: ASSIM
    PROCEDURE(IRUN), DEFERRED, PASS :: RUN
    PROCEDURE(IDEALLOC), DEFERRED, PASS :: DEALLOC

    PROCEDURE(IIO_READ), DEFERRED, PASS :: READSURF
    PROCEDURE(IIO_WRITE), DEFERRED, PASS :: WRITESURF
    PROCEDURE(IIO_WRITE), DEFERRED, PASS :: WRITE_DIAG

    PROCEDURE(IRESPONSE), DEFERRED, PASS :: GET_RESPONSE
    PROCEDURE(IDIAG), DEFERRED, PASS :: DIAG_MISC

    PROCEDURE, PASS :: BIND_INPUTS
    PROCEDURE, PASS :: COUPLING_ICEFLUX
END TYPE SEA_ICE_t
```

```
ABSTRACT INTERFACE
  SUBROUTINE IINIT(THIS, HPROGRAM)
    IMPORT :: SEA_ICE_t
    CLASS(SEA_ICE_t) :: THIS
    CHARACTER(LEN=6), INTENT(IN)  :: HPROGRAM
  END SUBROUTINE IINIT

  SUBROUTINE IPREP(THIS, <...>)
    ! ...
  END SUBROUTINE IPREP

  SUBROUTINE IASSIM(THIS, <...>)
  END SUBROUTINE IASSIM

  ! ...
  ! The rest of deferred procedures
  ! ...
END INTERFACE
```

This interface defines all actions that could be taken over a sea ice scheme

# Generic code on the calling side. Initialization

- Sea ice variable is defined as a generic pointer:

```
! modd_seafluxn.F90
  CLASS(SEA_ICE_t), POINTER :: ICE ! Sea-ice state
```

- The only place where the sea ice scheme should be checked in the classic way is allocation of the pointer with correct type:

```
! init_seafluxn.F90
SELECT CASE(SM%S%CSEAICE_SCHEME)
  CASE('GELATO')
    ALLOCATE(GELATO_t   :: SM%S%ICE)
  CASE('SICE  ')
    ALLOCATE(SICE_t     :: SM%S%ICE)
  CASE('NONE  ')
    ALLOCATE(ICE_NONE_t :: SM%S%ICE)
  CASE DEFAULT
    CALL ABOR1_SFX('Unknown sea ice scheme: ' // TRIM(SM%S%CSEAICE_SCHEME))
END SELECT
```

- After this stage all interactions with the sea ice scheme should be done through the generic interface

# Generic code on the calling side. Interaction with a scheme through the generic interface

## Classic SURFEX approach

```
! coupling_seafluxn.F90
IF (S%CSEAICE_SCHEME=='GELATO') THEN
  CALL SEAICE_GELATO1D_n(S, HPROGRAM,PTIMEC, PTSTEP)
END IF
```

```
! diag_inline_seafluxn.F90
IF (DGMSI%LDIAG_MISC_SEAICE) THEN
   IF (TRIM(S%CSEAICE_SCHEME) == 'GELATO') THEN
      GELATO_DIM=SIZE(PTA)
      DGMSI%XSIT  = RESHAPE( &
        glt_avhicem(S%TGLT%dom,S%TGLT%sit), &
        (/GELATO_DIM/))
      DGMSI%XSND  = RESHAPE( &
        glt_avhsnwm(S%TGLT%dom,S%TGLT%sit), &
        (/GELATO_DIM/))
      DGMSI%XMLT  = S%TGLT%oce_all(:,1)%tml
   ELSE
      ! Placeholder for an alternate seaice scheme
   END IF
END IF
```

```
! writesurf_seaicen.F90
IF (S%CSEAICE_SCHEME == 'GELATO') THEN
   YCOMMENT='Number of sea-ice layers'
   CALL WRITE_SURF(HSELECT,HPROGRAM,'ICENL',nl,IRESP,YCOMMENT)
   ! ... and the rest of GELATO-specific calls...
END IF
```

## Object-oriented approach

```
! coupling_seafluxn.F90
CALL S%ICE%RUN(<...>)
```

```
! diag_inline_seafluxn.F90
! GELATO-specific code has been moved to the
! source file implementing the common interface
! for GELATO model
IF (DGMSI%LDIAG_MISC_SEAICE) &
  CALL S%ICE%DIAG_MISC(DGMSI)
```

```
! writesurf_seaicen.F90
CALL S%ICE%WRITESURF(HSELECT, HPROGRAM)
```

# Note on the `CSEAICE_SCHEME==NONE` option

- Originally in SURFEX `'NONE'` indicates no prognostic sea ice scheme
- As result only the common diagnostic routine is called
- But when introducing a generic interface it is called unconditionally

# Note on the `CSEAICE_SCHEME==NONE` option

- Originally in SURFEX `'NONE'` indicates no prognostic sea ice scheme
- As result only the common diagnostic routine is called
- But when introducing a generic interface it is called unconditionally

As a consequence the `'NONE'` scheme should also fully implement the generic interface

# Note on the `CSEAICE_SCHEME==NONE` option

- Originally in SURFEX `'NONE'` indicates no prognostic sea ice scheme
- As result only the common diagnostic routine is called
- But when introducing a generic interface it is called unconditionally

As a consequence the `'NONE'` scheme should also fully implement the generic interface

Since this scheme is purely diagnostic most of the routines are just empty stubs

# Note on the `CSEAICE_SCHEME==NONE` option

- Originally in SURFEX `'NONE'` indicates no prognostic sea ice scheme
- As result only the common diagnostic routine is called
- But when introducing a generic interface it is called unconditionally

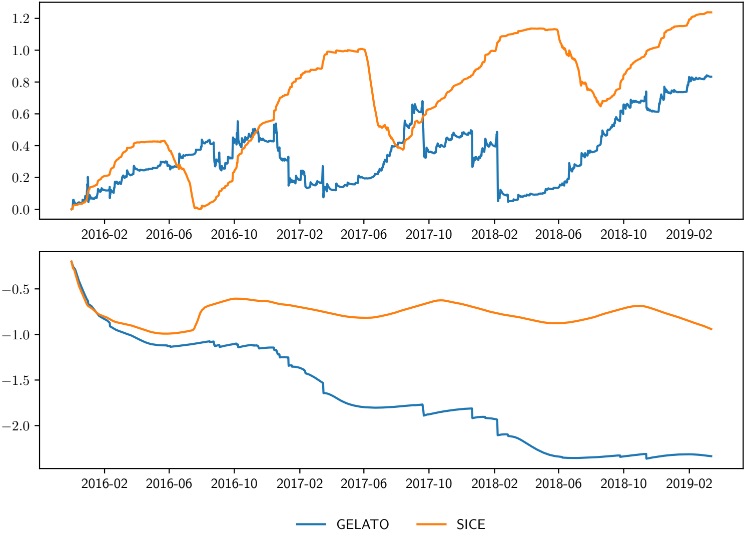
As a consequence the `'NONE'` scheme should also fully implement the generic interface

Since this scheme is purely diagnostic most of the routines are just empty stubs

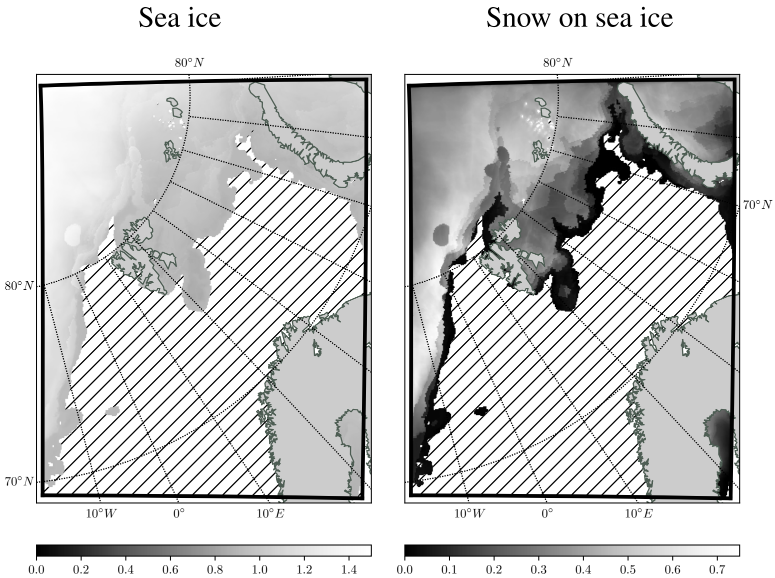
These unnecessary calls introduce some overhead compared to the original SURFEX code

# Example of SICE and GELATO performance

Test SURFEX off-line single point experiment



# Performance of SICE in the operational system, February 14th



Sea ice

Snow on sea ice

# Developments towards the data assimilation in SICE

- SICE is utilized in operational environment but runs freely
- Sea ice cover is defined by SIC from an external source
- But lack of ice dynamics affects the performance

# Developments towards the data assimilation in SICE

- SICE is utilized in operational environment but runs freely
- Sea ice cover is defined by SIC from an external source
- But lack of ice dynamics affects the performance

Some of the problems could be overcome by using data assimilation

But available data mainly consist of remote sensing products

# Possible sea ice variables to be assimilated

- Sea ice thickness and snow water equivalent over sea ice
  Assimilating these variables could help to improve the sea ice state in absence of ice dynamics. But most of the products are highly uncertain.
- Sea ice temperature
  Would help to improve the sea ice surface temperatures for a first few hours of forecast. Though, generally less beneficial than the above-mentioned.

# Possible sea ice variables to be assimilated

- Sea ice thickness and snow water equivalent over sea ice
  Assimilating these variables could help to improve the sea ice state in absence of ice dynamics. But most of the products are highly uncertain.
- Sea ice temperature
  Would help to improve the sea ice surface temperatures for a first few hours of forecast. Though, generally less beneficial than the above-mentioned.

The ALERTNESS project has a task that aims to improve representation of the sea ice cover by assimilating an ice surface temperature product.

# But to add a new DA scheme to SURFEX is not an easy task

- The original idea is to use EKF within the SODA framework
- But SODA is highly tied to the ISBA DA
- Same holds for the (S)EKF source code except some auxiliary routines

# SICE data assimilation framework

Data assimilation code for SICE is fully encapsulated within the SICE source code

Calling side does not interact with SICE-specific components

```fortran
! assim_sean.F90
ZSIC(:) = PSIC_IN(:)
! Consistency check
WHERE(ABS(ZSIC(:)) > 0)
  WHERE( ZSIC(:) < 0.01 ) ZSIC(:) = 0.0
ENDWHERE
! Main generic driver for the sea ice DA code
CALL S%ICE%ASSIM(HPROGRAM, ZSIC, PLON_IN, PLAT_IN)

IF(S%LHANDLE_SIC .AND. (S%CSEAICE_SCHEME == 'SICE  ')) THEN
  S%XSIC(:) = ZSIC(:)
END IF
```

All DA-related IO is performed by the SICE DA routine (unlike the standard approach when IO is handled on the higher level)

# Bias-aware Kalman filter

Classic Kalman filter is designed for unbiased control variables

$$\begin{aligned}
\mathrm{B} &= \mathrm{MAM}^{\mathrm{T}} + \mathrm{Q} \\
\mathrm{K} &= \mathrm{BH}^{\mathrm{T}}(\mathrm{HBH}^{\mathrm{T}} + \mathrm{R})^{-1} \\
X_a &= X_b + \mathrm{K}\,(Y - \mathcal{H}\,(X_b)) \\
\mathrm{A} &= (\mathrm{I} - \mathrm{KH})\mathrm{B}(\mathrm{I} - \mathrm{KH})^{\mathrm{T}} + \mathrm{KRK}^{\mathrm{T}}
\end{aligned}$$

But ice surface temperature in SICE (esp. in the snow-free configuration) is liable to systematic model errors

# Bias-aware Kalman filter

Model bias could be accounted by extending Kalman filter formulations

$$
\begin{aligned}
\mathrm{B} &= \mathrm{MAM}^{\mathrm{T}} + \mathrm{Q} \\
\mathrm{K}^{b} &= \mathrm{B}_{b}\mathrm{H}^{\mathrm{T}}(\mathrm{HB}_{b}\mathrm{H}^{\mathrm{T}} + \mathrm{HBH}^{\mathrm{T}} + \mathrm{R})^{-1} \\
\mathrm{K} &= \mathrm{BH}^{\mathrm{T}}(\mathrm{HBH}^{\mathrm{T}} + \mathrm{R})^{-1} \\
b_a &= b_b - \mathrm{K}^{b}(Y - \mathcal{H}(X_b - b_b)) \\
X_a &= (X_b - b_a) + \mathrm{K}(Y - \mathcal{H}(X_b - b_a)) \\
\mathrm{A} &= (\mathrm{I} - \mathrm{KH})\mathrm{B}(\mathrm{I} - \mathrm{KH})^{\mathrm{T}} + \mathrm{KRK}^{\mathrm{T}}
\end{aligned}
$$

# Development status of EKF for SICE

The main framework is implemented for snow-free and snow-covered states of the ice surface

Initial tests with idealized data to study the behaviour of the DA scheme

Off-line experiments suggest that bias correction should be applied as a correction term during the forecast

Questions?