

---

# **seaice Documentation**

***Release 0.1b***

**Anders Damsgaard**

**Apr 13, 2017**



**CONTENTS:**

<b>1</b>	<b>The <code>seaice</code> module</b>	<b>1</b>
<b>2</b>	<b>Indices and tables</b>	<b>7</b>
	<b>Python Module Index</b>	<b>9</b>
	<b>Index</b>	<b>11</b>



## THE SEAICE MODULE

SeaIce

**Package name** seaice

**Release date** 2017-04-13

**Authors** Anders Damsgaard:

**URL** <https://github.com/anders-dc/seaice>

**License** GPLv3

```
class seaice.IceFloeCylindrical(lin_pos, thickness, contact_radius, areal_radius=None,
                               lin_vel=[0.0, 0.0], lin_acc=[0.0, 0.0], force=[0.0, 0.0],
                               ang_pos=0.0, ang_vel=0.0, ang_acc=0.0, torque=0.0, den-
                               sity=934.0, rotating=True, fixed=False)
```

Cylindrical ice floe object.

#### Parameters

- **lin\_pos** (*list or numpy.array*) – Floe linear position [m]
- **thickness** (*float*) – Floe thickness [m]
- **contact\_radius** (*float*) – Floe radius during interactions [m]
- **areal\_radius** (*float*) – Floe areal radius on the sea surface [m]. If not set, this parameter will equal the *contact\_radius* value
- **lin\_vel** (*list or numpy.array*) – Floe linear velocity [m/s]
- **lin\_acc** (*list or numpy.array*) – Floe linear acceleration [m/s<sup>2</sup>]
- **force** (*list or numpy.array*) – Sum of forces [N]
- **ang\_pos** (*float*) – Floe angular position [rad]
- **ang\_vel** (*float*) – Floe angular velocity [rad/s]
- **ang\_acc** (*float*) – Floe angular acceleration [rad/s<sup>2</sup>]
- **torque** (*float*) – Sum of forces [N]
- **density** (*float*) – Floe density [kg/m<sup>3</sup>]
- **rotating** (*bool*) – The floe is free to rotate
- **fixed** (*bool*) – The floe is free to move linearly and/or rotationally

**find\_accelerations()**

Determine current linear and angular accelerations based on the current sum of forces.

**mass()**

Determine the current floe mass.

**Returns** The floe mass based on its areal radius, thickness, and density.

**Return type** float

**moment\_of\_inertia\_vertical()**

Determines the rotational moment of inertia for rotation around the floe center with a vertical rotation axis.

**Returns** The vertical rotational moment of inertia

**Return type** float

**surface\_area()**

Determine the current floe surface area.

**Returns** The floe mass based on its areal radius.

**Return type** float

**update\_position(dt, method='TY3')**

Update the kinematics through explicit temporal integration using a third-order Taylor expansion.

**Parameters**

- **dt** (*float*) – The time step length
- **method** – The integration method to choose ('TY2' or 'TY3' (default))

**update\_position\_TY2(dt)**

Update the kinematics through explicit temporal integration of Newton's second law using a second-order Taylor expansion.

**Parameters** **dt** (*float*) – The time step length

**update\_position\_TY3(dt)**

Update the kinematics through explicit temporal integration of Newton's second law using a third-order Taylor expansion.

**Parameters** **dt** (*float*) – The time step length

**volume()**

Determine the current floe volume.

**Returns** The floe mass based on its areal radius, thickness.

**Return type** float

**class seaice.SquareGrid(nx, ny, dx=None, dy=None, Lx=None, Ly=None)**

A two-dimensional, orthogonal and Cartesian grid, with options to add field values at the center, edges, or corners.

**Parameters**

- **nx** (*int*) – The number of grid cells along x.
- **ny** (*int*) – The number of grid cells along y.

**getCenterCoordinate(i, j)**

Returns the center coordinate for the center of the cell with index *i* and *j*, sometimes referred to as the h-point.

**Parameters**

- **i** (*int*) – Cell index along x.

- `j (int)` – Cell index along *y*.

**Returns** The Cartesian coordinate for the cell center.

**Return type** `numpy.array`

See also: `getSouthFaceCoordinate()`, `getNorthFaceCoordinate()`,  
`getEastFaceCoordinate()`, `getWestFaceCoordinate()`, `getSouthWestCornerCoordinate()`,  
`getSouthEastCornerCoordinate()`, `getNorthWestCornerCoordinate()`, and  
`getNorthEastCornerCoordinate()`.

#### **getEastFaceCoordinate** (*i*, *j*)

Returns the east-oriented face-center coordinate for the cell with index *i* and *j*, sometimes referred to as the u-point.

##### **Parameters**

- `i (int)` – Cell index along *x*.
- `j (int)` – Cell index along *y*.

**Returns** The Cartesian coordinate for the center of the eastern cell face.

**Return type** `numpy.array`

See also: `getCenterCoordinate()`, `getSouthFaceCoordinate()`,  
`getNorthFaceCoordinate()`, `getWestFaceCoordinate()`,  
`getSouthWestCornerCoordinate()`, `getSouthEastCornerCoordinate()`,  
`getNorthWestCornerCoordinate()`, and `getNorthEastCornerCoordinate()`.

#### **getNorthEastCornerCoordinate** (*i*, *j*)

Returns the north-east oriented corner coordinate the cell with index *i* and *j*, sometimes referred to as the q-point.

##### **Parameters**

- `i (int)` – Cell index along *x*.
- `j (int)` – Cell index along *y*.

**Returns** The Cartesian coordinate for the center of the north-eastern cell corner.

**Return type** `numpy.array`

See also: `getCenterCoordinate()`, `getSouthFaceCoordinate()`,  
`getNorthFaceCoordinate()`, `getEastFaceCoordinate()`,  
`getWestFaceCoordinate()`, `getSouthWestCornerCoordinate()`,  
`getSouthEastCornerCoordinate()`, and `getNorthWestCornerCoordinate()`.

#### **getNorthFaceCoordinate** (*i*, *j*)

Returns the north-oriented face-center coordinate for the cell with index *i* and *j*, sometimes referred to as the v-point.

##### **Parameters**

- `i (int)` – Cell index along *x*.
- `j (int)` – Cell index along *y*.

**Returns** The Cartesian coordinate for the center of the northern cell face.

**Return type** `numpy.array`

See also: `getCenterCoordinate()`, `getSouthFaceCoordinate()`,  
`getEastFaceCoordinate()`, `getWestFaceCoordinate()`, `getSouthWestCornerCoordinate()`,

`getSouthEastCornerCoordinate()`, `getNorthWestCornerCoordinate()`, and `getNorthEastCornerCoordinate()`.

#### **getNorthWestCornerCoordinate** (*i*, *j*)

Returns the north-west oriented corner coordinate the cell with index *i* and *j*, sometimes referred to as the q-point.

##### **Parameters**

- *i* (*int*) – Cell index along *x*.
- *j* (*int*) – Cell index along *y*.

**Returns** The Cartesian coordinate for the center of the north-western cell corner.

**Return type** `numpy.array`

See also: `getCenterCoordinate()`, `getSouthFaceCoordinate()`, `getNorthFaceCoordinate()`, `getEastFaceCoordinate()`, `getWestFaceCoordinate()`, `getSouthWestCornerCoordinate()`, `getSouthEastCornerCoordinate()`, and `getNorthEastCornerCoordinate()`.

#### **getSouthEastCornerCoordinate** (*i*, *j*)

Returns the south-east oriented corner coordinate the cell with index *i* and *j*, sometimes referred to as the q-point.

##### **Parameters**

- *i* (*int*) – Cell index along *x*.
- *j* (*int*) – Cell index along *y*.

**Returns** The Cartesian coordinate for the center of the south-eastern cell corner.

**Return type** `numpy.array`

See also: `getCenterCoordinate()`, `getSouthFaceCoordinate()`, `getNorthFaceCoordinate()`, `getEastFaceCoordinate()`, `getWestFaceCoordinate()`, `getSouthWestCornerCoordinate()`, `getNorthWestCornerCoordinate()`, and `getNorthEastCornerCoordinate()`.

#### **getSouthFaceCoordinate** (*i*, *j*)

Returns the south-oriented face-center coordinate for the cell with index *i* and *j*, sometimes referred to as the v-point.

##### **Parameters**

- *i* (*int*) – Cell index along *x*.
- *j* (*int*) – Cell index along *y*.

**Returns** The Cartesian coordinate for the center of the southern cell face.

**Return type** `numpy.array`

See also: `getCenterCoordinate()`, `getNorthFaceCoordinate()`, `getEastFaceCoordinate()`, `getWestFaceCoordinate()`, `getSouthWestCornerCoordinate()`, `getSouthEastCornerCoordinate()`, `getNorthWestCornerCoordinate()`, and `getNorthEastCornerCoordinate()`.

#### **getSouthWestCornerCoordinate** (*i*, *j*)

Returns the south-west oriented corner coordinate the cell with index *i* and *j*, sometimes referred to as the q-point.

##### **Parameters**



- **i** (*int*) – Cell index along *x*.
- **j** (*int*) – Cell index along *y*.

**Returns** The Cartesian coordinate for the center of the south-western cell corner.

**Return type** `numpy.array`

See also: `getCenterCoordinate()`, `getSouthFaceCoordinate()`,  
`getNorthFaceCoordinate()`, `getEastFaceCoordinate()`,  
`getWestFaceCoordinate()`, `getSouthEastCornerCoordinate()`,  
`getNorthWestCornerCoordinate()`, and `getNorthEastCornerCoordinate()`.

#### **getWestFaceCoordinate** (*i*, *j*)

Returns the west-oriented face-center coordinate for the cell with index *i* and *j*, sometimes referred to as the u-point.

##### **Parameters**

- **i** (*int*) – Cell index along *x*.
- **j** (*int*) – Cell index along *y*.

**Returns** The Cartesian coordinate for the center of the western cell face.

**Return type** `numpy.array`

See also: `getCenterCoordinate()`, `getSouthFaceCoordinate()`,  
`getNorthFaceCoordinate()`, `getEastFaceCoordinate()`,  
`getSouthEastCornerCoordinate()`, `getNorthWestCornerCoordinate()`, and  
`getNorthEastCornerCoordinate()`.

#### **setSize** (*origo*=[0.0, 0.0], *dx*=None, *dy*=None, *Lx*=None, *Ly*=None)

Used to determine the spatial dimensions of the grid. The user may provide cell widths (*dx* and/or *dy*) or grid lengths (*Lx* and/or *Ly*). It is implied that the grid width equals the cell width multiplied by the number of cells along each dimension.

##### **Parameters**

- **origo** (`numpy.array`) – Shift the grid the value of this 2d vector.
- **dx** (*float*) – Cell width along *x*.
- **dy** (*float*) – Cell width along *y*.
- **Lx** – Grid width along *x*.
- **Ly** – Grid width along *y*.



## INDICES AND TABLES

- `genindex`
- `modindex`
- `search`



## PYTHON MODULE INDEX

### S

seaice, [1](#)



## INDEX

### F

find\_accelerations() (seaice.IceFloeCylindrical method),  
1

### G

getCenterCoordinate() (seaice.SquareGrid method), 2  
getEastFaceCoordinate() (seaice.SquareGrid method), 3  
getNorthEastCornerCoordinate() (seaice.SquareGrid  
method), 3  
getNorthFaceCoordinate() (seaice.SquareGrid method), 3  
getNorthWestCornerCoordinate() (seaice.SquareGrid  
method), 4  
getSouthEastCornerCoordinate() (seaice.SquareGrid  
method), 4  
getSouthFaceCoordinate() (seaice.SquareGrid method), 4  
getSouthWestCornerCoordinate() (seaice.SquareGrid  
method), 4  
getWestFaceCoordinate() (seaice.SquareGrid method), 5

### I

IceFloeCylindrical (class in seaice), 1

### M

mass() (seaice.IceFloeCylindrical method), 1  
moment\_of\_inertia\_vertical() (seaice.IceFloeCylindrical  
method), 2

### S

seaice (module), 1  
setSize() (seaice.SquareGrid method), 5  
SquareGrid (class in seaice), 2  
surface\_area() (seaice.IceFloeCylindrical method), 2

### U

update\_position() (seaice.IceFloeCylindrical method), 2  
update\_position\_TY2() (seaice.IceFloeCylindrical  
method), 2  
update\_position\_TY3() (seaice.IceFloeCylindrical  
method), 2

### V

volume() (seaice.IceFloeCylindrical method), 2