

McPIC

(version 1.0)

**DOCUMENTATION FOR THE MULTI-CATEGORY
PARTICLE-IN-CELL SEA ICE MODEL**

G.M. Flato

Department of Fisheries and Oceans
Institute of Ocean Sciences
Sidney, British Columbia

1994

**CANADIAN TECHNICAL REPORT OF
HYDROGRAPHY AND OCEAN SCIENCES
NO. 158**



Fisheries
and Oceans

Pêches
et Océans

Canada

Canadian Technical Report of Hydrography and Ocean Sciences

These reports contain scientific and technical information of a type that represents a contribution to existing knowledge but which is not normally found in the primary literature. The subject matter is generally related to programs and interests of the Ocean Science and Surveys (OSS) sector of the Department of Fisheries and Oceans.

Technical Reports may be cited as full publications. The correct citation appears above the abstract of each report. Each report will be abstracted in Aquatic Sciences and Fisheries Abstracts. Reports are also listed in the Department's annual index to scientific and technical publications.

Technical Reports are produced regionally but are numbered and indexed nationally. Requests for individual reports will be fulfilled by the issuing establishment listed on the front cover and title page. Out of stock reports will be supplied for a fee by commercial agents.

Regional and headquarters establishments of Ocean Science and Surveys ceased publication of their various report series as of December 1981. A complete listing of these publications and the last number issued under each title are published in the *Canadian Journal of Fisheries and Aquatic Sciences*, Volume 38: Index to Publications 1981. The current series began with Report Number 1 in January 1982.

Rapport technique canadien sur l'hydrographie et les sciences océaniques

Ces rapports contiennent des renseignements scientifiques et techniques qui constituent une contribution aux connaissances actuelles mais que l'on ne trouve pas normalement dans les revues scientifiques. Le sujet est généralement rattaché aux programmes et intérêts du service des Sciences et Levés océaniques (SLO) du ministère des Pêches et des Océans.

Les rapports techniques peuvent être considérés comme des publications à part entière. Le titre exact figure au-dessus du résumé du chaque rapport. Les résumés des rapports seront publiés dans la revue Résumés des sciences aquatiques et halieutiques et les titres figureront dans l'index annuel des publications scientifiques et techniques du Ministère.

Les rapports techniques sont produits à l'échelon régional mais sont numérotés et placés dans l'index à l'échelon national. Les demandes de rapports seront satisfaites par l'établissement auteur dont le nom figure sur la couverture et la page de titre. Les rapports épuisés seront fournis contre rétribution par des agents commerciaux.

Les établissements des Sciences et Levés océaniques dans les régions et à l'administration centrale ont cessé de publier leurs diverses séries de rapports depuis décembre 1981. Vous trouverez dans l'index des publications du volume 38 du *Journal canadien des sciences halieutiques et aquatiques*, la liste de ces publications ainsi que le dernier numéro paru dans chaque catégorie. La nouvelle série a commencé avec la publication du Rapport n° 1 en janvier 1982.

Canadian Technical Report of Hydrography
and Ocean Sciences No. 158

1994

McPIC
(version 1.0)

**Documentation for the Multi-category Particle-In-Cell
Sea Ice Model**

by

G. M. Flato

*Institute of Ocean Sciences
Sidney, BC*

present address for G. M. Flato:
Canadian Climate Centre
University of Victoria
Victoria, B. C. Canada

© Minister of Supply and Services Canada 1994
Cat. No. 97-18/158 ISSN 0711-6764

Correct citation for this publication:

Flato, G. M. 1994. McPIC - Documentation for the Multi-category Particle-In-Cell
Sea Ice Model. Can. Tech. Rep. Hydrogr. Ocean Sci. 158: 74 p.

ABSTRACT

Flato, G. M. 1994. McPIC - Documentation for the Multi-category Particle-In-Cell Sea Ice Model.
Can. Tech. Rep. Hydrogr. Ocean Sci. 158: 74 p.

A multi-category particle-in-cell sea ice model is described. The objective of this model is to improve the accuracy of operational ice edge forecasts by reducing the diffusion inherent in conventional models. This is achieved by Lagrangian treatment of ice area and volume conservation equations in which particles, each representing a collection of ice floes or a region of ice-covered area, are advected individually. The velocity field used in this advection is obtained by solving the sea ice momentum equation on a fixed Eulerian grid. The volume of ice represented by each particle is distributed amongst several fixed thickness categories; the distribution evolves over time as a result of ice ridging during deformation and thermodynamic growth or melt. The thermodynamic component of the model includes a surface energy budget and heat conduction calculation for each thickness category and a fixed-depth mixed-layer beneath the ice which acts as a thermal reservoir under ice-free conditions. The theory and computer code are described along with an example calculation.

Keywords: sea ice modelling

RÉSUMÉ

Flato, G. M. 1994. McPIC - Documentation for the Multi-category Particle-In-Cell Sea Ice Model.
Can. Tech. Rep. Hydrogr. Ocean Sci. 158: 74 p.

Un modèle de glace de mer à catégorie multiple, représentés par des particules cellulaires est décrit. Le but de ce modèle est d'améliorer la précision des prévisions opérationnelles de la limite de glace en réduisant la diffusion inhérente dans les modèles conventionnels. Cela est réalisé par un traitement Lagrangien des équations de conservation de volume et de surface de la glace dans lequelles des particules, chacune représentant une accumulation de glaces flottantes ou une région couverte de glace, sont advectées individuellement. Le champ de vitesse utilisé pour cette advection est obtenu en résolvant l'équation du mouvement de la glace de mer sur une grille Eulerienne fixe. Le volume de glace représenté par chaque particule est distribué parmi plusieurs catégories d'épaisseur; la distribution évolue dans le temps suivant le processus de formation de crêtes de glace ainsi que suivant la croissance ou la fonte thermodynamiques. La composante thermodynamique du modèle comprend un budget d'énergie à la surface, un calcul de la conduction de la chaleur pour chaque catégorie d'épaisseur, et une couche d'eau mixte à profondeur fixe sous la glace qui agit comme réservoir thermique lors de conditions sans glace. La théorie et les codes informatiques sont décrits ainsi qu'un exemple de calcul.

Mots-clefs: modélisation de la glace de mer

Table of Contents

Introduction	1
Background Theory.....	2
Overview of particle-in-cell method	2
Interpolation details.....	4
Momentum equation	5
Thickness distribution	6
Redistribution function	7
Finite-difference implementation.....	8
Thermodynamics.....	12
Discussion of Parameterizations	16
References	18
Acknowledgements	18
Appendix 1 — McPIC Code Listing.....	19
Appendix 2 — Detailed Description of McPIC Code	56
Appendix 3 — McPIC - Test Case Forcing and Output Plots	63

Introduction

The McPIC sea ice model was developed at the Institute of Ocean Sciences as part of a PERD-funded project on ice-ocean modelling of the Beaufort Sea. The objective of the model is to improve the accuracy of ice edge forecasts by reducing the ambiguity in definition and artificial diffusion of the ice edge in conventional Eulerian grid models. This is accomplished by introducing a Lagrangian treatment of the ice volume and area conservation equations via the 'particle-in-cell' method. Fundamental to this method is the partitioning of conserved scalar quantities into individual particles, each of which is advected separately. The velocity field responsible for particle motion is calculated on a fixed, regularly-spaced grid, from which the velocity of a particle is obtained by interpolation. Scalar quantities carried by the particles are interpolated onto the fixed grid to allow solution of the momentum equations. The basic scheme and some example calculations were presented by Flato (1993a, 1993b); however, the model described in this documentation contains modifications not included in the published version. The principal difference is the addition of a multi-category thickness distribution, based on the theory of Thorndike et al. (1975) and similar in principle to the multi-level model of Hibler (1980) and Flato and Hibler (1994).

In the following section, the particle-in-cell method and the thickness distribution theory will be briefly reviewed. The notation has been changed slightly from that in the papers listed above, both for uniformity and to make the correspondence between equations in this document and the code as clear as possible. Also included is a discussion regarding choices made in the parameterization of ice strength and ridge redistribution.

A code listing is provided in Appendix 1, followed by a detailed description of the code in Appendix 2. The example calculation included in the code is briefly discussed in Appendix 3. This code with example is available by anonymous ftp from 'ftp.ios.bc.ca'.

This model is a research tool in the early stages of development. It is available for use by others, but those wishing to do so should check the code carefully and be sure that the assumptions and parameter choices are appropriate for their particular application.

Background Theory

Overview of particle-in-cell method

The PIC method was originally designed to solve fluid dynamics problems involving shocks or free surfaces (Harlow, 1964) and is now widely used in plasma physics. Essentially, the method involves partitioning a continuous state variable into discrete particles, each of which is advected in a Lagrangian sense, while the momentum equation is solved on a fixed reference grid. In the present context, consider ice-covered area in some region partitioned into N particles, the area of the n^{th} particle at time t being $A(n,t)$, and its position being $\mathbf{X}(n,t)$.¹ [The notation here is that \mathbf{X} indicates the position of a particle, whereas \mathbf{x} is the location in a fixed reference grid.] Given a continuous velocity field, $\mathbf{u}(\mathbf{x},t)$, advection is described formally by

$$\mathbf{X}(n, t + \Delta t) = \mathbf{X}(n, t) + \int_{t}^{t+\Delta t} \mathbf{u}(\mathbf{X}(n, t'), t') dt' \quad (1)$$

where Δt is the time step and the integral in (1) is understood to be an integral along the particle trajectory.

The corresponding state variable, concentration, $a(\mathbf{x},t)$, can be defined as an average of the particles' ice-covered area per unit total area. To facilitate this averaging, and subsequently to solve the momentum equations, we define a fixed, uniformly spaced Eulerian grid underlying the domain with the state variables defined at the centres of the grid cells, denoted $\mathbf{x}_{i,j}$, as shown in Figure 1. Various interpolation schemes can be used to project from the discrete particle representation onto a fixed grid. These can be written generally as

$$a(\mathbf{x}_{i,j}, t) = \sum_{n=1}^N \omega(\mathbf{x}_{i,j}, \mathbf{X}(n, t)) A(n, t) \frac{1}{\Delta x \Delta y} \quad (2)$$

where ω is some weighting function to be described shortly, and Δx and Δy are the grid spacing in the x and y directions. Equations (1) and (2) represent the advection of a scalar, ice-covered area. However, an additional step is required to insure that the concentration never exceeds unity. The physical process involved here is the destruction of thin ice to form thick ridged ice during convergent deformation. The details of this process will be discussed momentarily, but the overall constraint is that a must be less than or equal to one. To enforce this constraint we must reduce the individual particle areas, $A(n,t)$, in such a way that the interpolated field of a from equation (2) is everywhere less than or equal to one. An exact calculation would in general require an iterative scheme since, depending on the form of ω , a given particle may influence the

¹Plain text variables indicate scalar quantities while bold-faced variables indicate vectors.

concentration in several surrounding grid cells. Instead, we will use an approximate procedure whereby the area of a particle is reduced by the amount necessary to insure $a \leq 1$ for all the grid cells it influences.

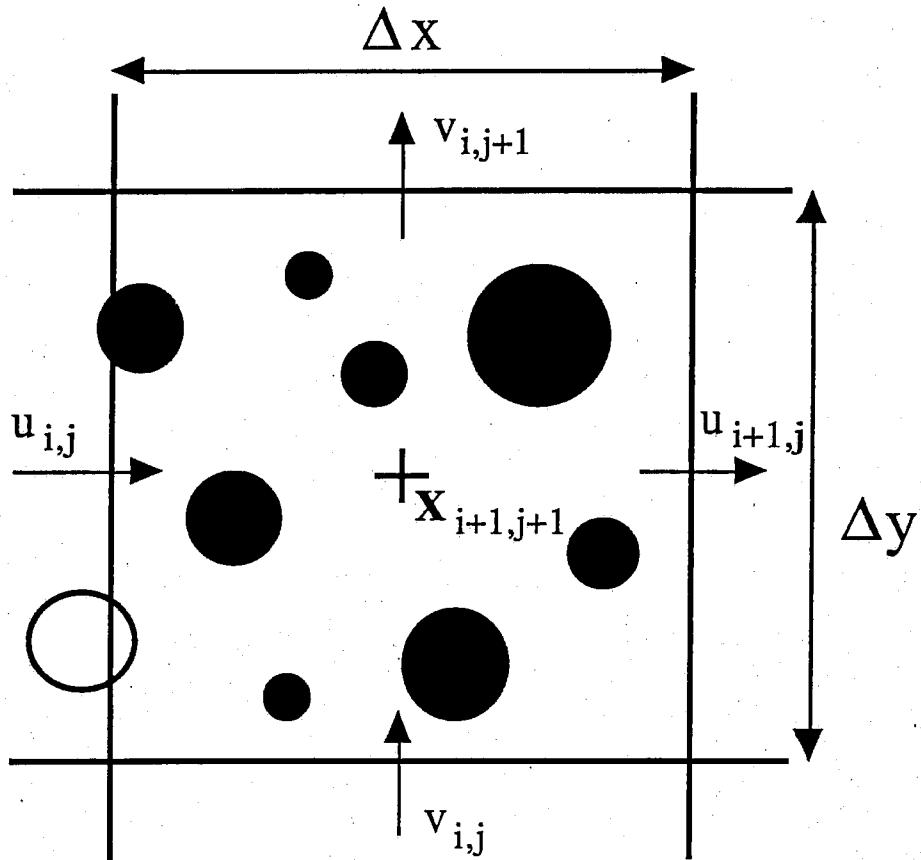


Figure 1. Sketch of grid cell showing scalar location, $x_{i+1,j+1}$, velocity components, $u_{i,j}$ and $v_{i,j}$, and example particles.

The modified particle areas are given by

$$\tilde{A}(n,t + \Delta t) = \frac{A(n,t)}{\max_{i,j} \left\{ 1, a(x_{i,j}, t) [1 - \delta(\omega(x_{i,j}, X(n,t)))] \right\}} \quad (3)$$

where $\tilde{A}(n,t + \Delta t)$ are the corrected values of $A(n,t)$ and δ is the Dirac delta function: $\delta(x)=0$ if $x \neq 0$; $\delta(x)=1$ if $x=0$. This approximate scheme has the disadvantage that it may reduce concentration values which were already less than one, however this error is small.

Interpolation details

A number of interpolation schemes could be used in equation (2). Pavia and Cushman-Roisin (1988), in a frontal geostrophic ocean model, used a scheme in which the particle volume was distributed amongst the nearest 9 grid points. This smoothed the granularity or noise associated with the discrete particle representation and enhanced the computational stability in their application (in which the momentum balance was determined entirely by spatial gradients in layer thickness). In the ice model considered here, spatial gradients in mean thickness and concentration enter the momentum balance rather weakly and so, to avoid excessive smoothing, a simpler bilinear interpolation scheme was used, in which the particle volume is distributed among the four surrounding grid cells, viz.

$$\omega(\mathbf{x}_{i,j}, \mathbf{X}(n,t)) = [\Delta x - |X(n,t) - x_{i,j}|][\Delta y - |Y(n,t) - y_{i,j}|] \frac{S(i,j,n)}{\Delta x \Delta y} \quad (4)$$

$$S(i,j,n) = \begin{cases} 1 & \text{if } |X(n,t) - x_{i,j}| \leq \Delta x \text{ and } |Y(n,t) - y_{i,j}| \leq \Delta y \\ 0 & \text{otherwise} \end{cases}$$

where $x_{i,j}$ and $y_{i,j}$ are the x - and y -components of $\mathbf{x}_{i,j}$, and X and Y are x - and y -components of \mathbf{X} .

Returning now to the particle advection described by (1), we note that in practice the velocity field is not a continuous function of \mathbf{x} , but rather is known only at discrete grid points. Therefore, interpolation from the grid velocities to the particle location is required — in essence, a reverse of the interpolation involved in equation (2). In order to use the same interpolation scheme for velocity as for thickness and concentration, the velocity components in the staggered grid used here (Figure 1) are averaged leading to

$$u(\mathbf{X}(n,t)) = \frac{1}{2} \sum_{i=1}^I \sum_{j=1}^J \omega\left(\mathbf{x}_{i+\frac{1}{2}, j+\frac{1}{2}}, \mathbf{X}(n,t)\right) (u_{i,j} + u_{i,j-1}) \quad (5)$$

$$v(\mathbf{X}(n,t)) = \frac{1}{2} \sum_{i=1}^I \sum_{j=1}^J \omega\left(\mathbf{x}_{i+\frac{1}{2}, j+\frac{1}{2}}, \mathbf{X}(n,t)\right) (v_{i,j} + v_{i-1,j})$$

where I and J are the number of grid cells in the x - and y -directions respectively, and u and v are the x - and y -components of \mathbf{u} . The integral in equation (1) is approximated simply as

$$\int_t^{t+\Delta t} \mathbf{u}(\mathbf{X}(n,t'), t') dt' = \mathbf{u}(\mathbf{X}^*, t) \Delta t \quad (6)$$

$$\mathbf{X}^* = \mathbf{X}(n,t) + \mathbf{u}(\mathbf{X}(n,t), t) \frac{\Delta t}{2}$$

Note that two applications of the velocity interpolation are required in (6). Clearly, there are more elaborate schemes available to evaluate this integral; however for the relatively short time steps used here, this one proved satisfactory.

Momentum equation

The sea ice momentum equation is (Hibler, 1979)

$$m \frac{d\mathbf{u}}{dt} = \tau_a + \tau_w - m\hat{f}\mathbf{k} \times \mathbf{u} - mg\nabla H + \nabla \cdot \underline{\sigma} \quad (7)$$

where: m is the ice mass per unit area; $d/dt = \partial/\partial t + \mathbf{u} \cdot \nabla$ is the substantial derivative; \mathbf{u} is the ice velocity vector; \hat{f} is the Coriolis parameter; \mathbf{k} is a unit vertical vector, g is the acceleration of gravity; H is the sea surface elevation, and $\underline{\sigma}$ is the two-dimensional internal ice stress tensor.

The latter is obtained by assuming a constitutive law governing the large-scale behavior of the ice pack — a further discussion of this can be found in Ip *et al.* (1991). τ_a and τ_w are the wind and water drag terms, which appear as body forces in (1), and are expressed as

$$\tau_a = C_a (\mathbf{U}_g \cos \phi + \mathbf{k} \times \mathbf{U}_g \sin \phi) \quad (8)$$

$$\tau_w = C_w ((\mathbf{U}_w - \mathbf{u}) \cos \theta + \mathbf{k} \times (\mathbf{U}_w - \mathbf{u}) \sin \theta)$$

where \mathbf{U}_g is the geostrophic wind vector, \mathbf{U}_w is the geostrophic current vector, C_a and C_w are wind and water drag coefficients which may be constants (linear drag) or functions of wind or current speed, and ϕ and θ are wind and water drag turning angles. The momentum equation, (7), is further simplified by neglecting the inertial and non-linear momentum advection terms. This is justified by a scaling argument given a typical ice velocity of $\sim 0.05 \text{ m s}^{-1}$, ice thickness of $\sim 2 \text{ m}$, grid spacing of 10 km, time step of 2 hr, and wind drag of $\sim 0.1 \text{ N m}^{-2}$.

The internal ice stress term, $\nabla \cdot \underline{\sigma}$, is obtained by assuming a Mohr-Coulomb rheology in which the shear strength is proportional to the pressure with an internal friction angle of 30° . This rheology is implemented using a slightly modified version of the scheme presented in Flato and Hibler (1992). The maximum pressure, beyond which compressive failure occurs, is parameterized in terms of the mean thickness and concentration (Hibler, 1979) as

$$p_{max} = P^* h \exp(-K(1-a)) \quad (9)$$

where p_{max} is the maximum pressure, P^* is the large-scale uniaxial compressive strength taken to be 27.5 kPa, h is the mean ice thickness (ice volume per unit area) and $K=20$ is an empirical constant. More details about this rheology and the numerical scheme used to implement it can be found in Flato and Hibler (1992) and Ip *et al.* (1991).

Thickness distribution

The initial implementation of a PIC sea ice model (Flato, 1993a; 1993b) had each particle representing only two state variables, concentration and mean thickness (corresponding to particle area and volume). McPIC, on the other hand, makes use of the thickness distribution theory of Thorndike *et al.* (1975) so that each particle represents a distribution of ice thicknesses. That is, a particle represents an ice-covered area, $A(n,t)$, within which the probability of occurrence of ice with thickness between h and $h+dh$ is defined as $g(h, n, t)$ — the so-called thickness distribution function [subsequently, where there is no ambiguity, we will write $g(h, n, t)$ simply as $g(h)$]. An important point to note is that, since a particle represents only ice-covered area, open water is not included in the particle's $g(h)$, i.e. $g(0)\equiv 0$. Therefore,

$$\int_0^\infty g(h)dh = \lim_{\epsilon \rightarrow 0} \int_\epsilon^\infty g(h)dh = 1 \quad (10)$$

For a given particle, the area occupied by ice with thickness between h_1 and h_2 is given by

$$A_{h_1, h_2}(n, t) = \int_{h_1}^{h_2} g(h, n, t)dh A(n, t) \quad (11)$$

and the ice volume associated with a given particle is

$$V(n, t) = \int_0^\infty g(h, n, t)h dh A(n, t) \quad (12)$$

Following Thorndike *et al.* (1975), the evolution of $g(h)$ is described by

$$\frac{\partial g}{\partial t} = -g\nabla \cdot \mathbf{u} - \frac{\partial}{\partial h}(fg) + \Psi \quad (13)$$

The first term on the RHS represents area change due to deformation (note that the advective term, $\mathbf{u} \cdot \nabla g$, which appears in the Eulerian grid version of (13) does not appear in the Lagrangian particle version). The second term on the RHS represents changes in $g(h)$ due to the thermodynamic growth or melt rate, $f(h)$. The final term, Ψ , is the ridge redistribution function which, following Thorndike *et al.* (1975), must satisfy two constraints:

$$\int_0^\infty \Psi dh = \nabla \cdot \mathbf{u} \quad (14)$$

$$\int_0^\infty h\Psi dh = 0 \quad (15)$$

The first of these constraints insures area conservation while the second insures volume conservation (that is, ridging does not destroy or create ice volume, it only redistributes volume between thickness categories — an implicit assumption is that ridged ice is pore-free).

In the PIC context, we don't have $\nabla \cdot \mathbf{u}$ defined for an individual particle; however, we do have the change in area of a particle over a time step (eq. 3) and so an appropriate definition of $\nabla \cdot \mathbf{u}$ in (13) and (14) is

$$\nabla \cdot \mathbf{u} = \frac{\tilde{A}(n, t + \Delta t) - A(n, t)}{\tilde{A}(n, t + \Delta t) \Delta t} \quad (16)$$

where $\tilde{A}(n, t + \Delta t)$ is the 'deformed' particle area, which differs from $A(n, t)$ only under convergent conditions.

Redistribution function

The initial version of this model assumes that ridging occurs only as a result of convergence — no ridges are formed during pure shear (see Discussion of Parameterizations below). In this case, the redistribution function is defined as (Thorndike *et al.*, 1975)

$$\Psi = \omega_r(h, g) |\nabla \cdot \mathbf{u}| \quad (17)$$

where

$$\omega_r(h, g) = \frac{-a(h) + n(h)}{-\int_0^h [-a(h) + n(h)] dh} \quad (18)$$

The first term in the numerator² is the distribution of thin ice destroyed by ridging while the second term is the distribution of newly-formed ridged ice. The denominator is required to satisfy constraint (14).

The distribution of thin ice destroyed by ridging is obtained by weighting the distribution of existing ice such that the thinnest ice available is preferentially ridged. Following Thorndike *et al.* (1975), we choose this weighting function such that only a fixed fraction of the ice area is available to participate in ridging, viz.

$$a(h) = b(h)g(h) \quad (19)$$

where

$$b(h) = \begin{cases} \frac{2}{G^*} \left[1 - \frac{G(h)}{G^*} \right] & ; \quad 0 \leq G(h) \leq G^* \\ 0 & ; \quad G(h) > G^* \end{cases} \quad (20)$$

² The notation here is a bit awkward: $a(h)$ is the distribution of thin ice destroyed by ridging (to be consistent with the notation of Thorndike *et al.*), while $a(x_{i,j}, t)$ is the concentration in the fixed reference grid. It should be clear from the context which usage is intended.

G^* is a fixed fraction (taken here to be 0.15 — this implies that only the thinnest 15% of the ice-covered area is available for ridging), and $G(h)$ is the cumulative thickness distribution function

$$G(h) = \int_0^h g(h') dh' \quad (21)$$

The distribution of ridged ice created during a deformation event, $n(h)$, is determined by a ridge redistribution process, γ , through a convolution integral

$$n(h) = \int_0^h \gamma(h', h) a(h') dh' \quad (22)$$

Constraint (15) is satisfied by requiring that

$$\int_0^h h \gamma(h', h) dh' = h' \quad (23)$$

As explained in the Discussion of Parameterizations below, we make the assumption that ridged ice is a fixed multiple, k , of the thickness of the ice destroyed by ridging and hence

$$\gamma(h', h) = \frac{1}{k} \delta(h - kh') \quad (24)$$

where δ is the usual Dirac delta-function. Note that (24) satisfies (23).

Finite-difference implementation

The finite-difference implementation of the above redistribution scheme follows closely that of Hibler (1980). We define a fixed, staggered thickness grid as shown in Figure 2, in which the thickness category boundaries are denoted h_l^b and the thickness category centres are denoted h_l^c . The subscript, l ranges from 1 to NL , the number of thickness categories, for quantities defined at the grid centres, and from 1 to $NL+1$ for quantities defined at the grid boundaries. We also define a discrete thickness distribution function, \tilde{g}_l , such that

$$\tilde{g}_l = \int_{h_l^b}^{h_{l+1}^b} g(h) dh \quad ; \quad l = 1, NL \quad (25)$$

It is further assumed that $g(h)$ is constant over a thickness category so that $G(h)$ is piece-wise linear

$$G_l = G_{l-1} + \tilde{g}_{l-1} \quad ; \quad l = 2, NL+1 \quad (26)$$

$$G_1 = 0$$

We can now construct the discrete version of the weighting function, $b(h)$, which follows directly from (20), (26) and Figure 3; however, for algebraic reasons, application of the criterion

that $b(h)=0$ for $G(h)>G^*$ will be delayed until $a(h)$ is calculated from (19). Defined at the thickness category boundary points (labelled ' o_i ' in Figure 2), the discrete weighting function, b_l is given by

$$b_l = b_{l-1} - \frac{2[G_l - G_{l-1}]}{(G^*)^2} ; \quad l = 2, NL + 1 \quad (27)$$

$$b_1 = \frac{2}{G^*}$$

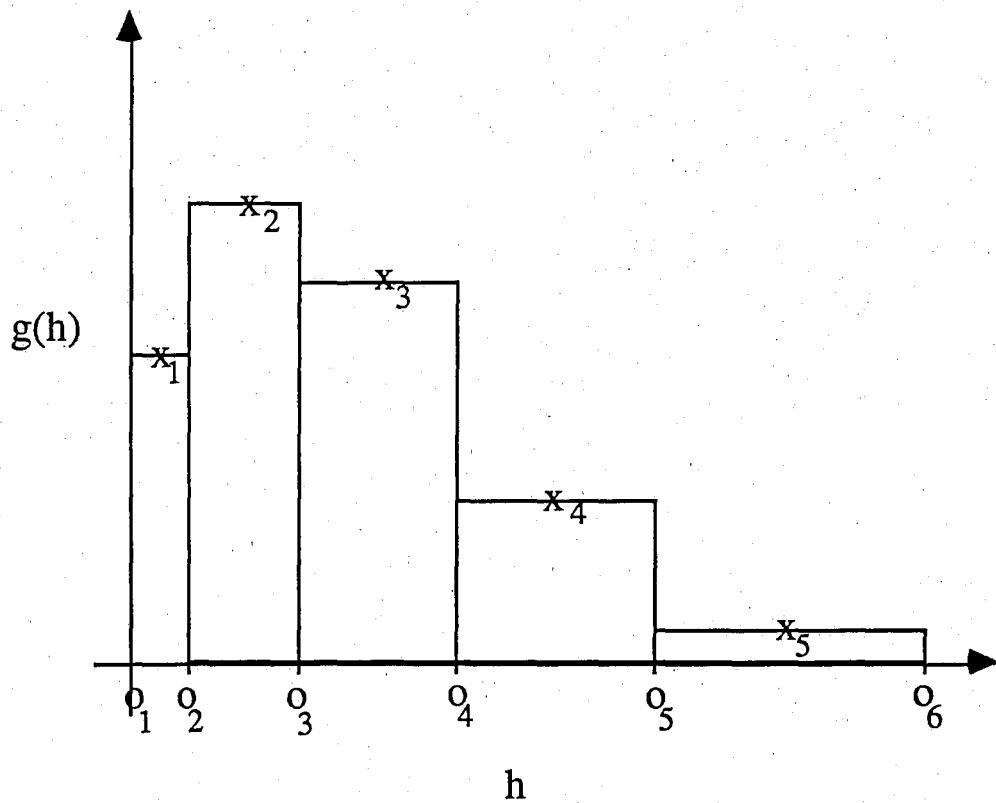


Figure 2. Sketch of finite-difference representation of thickness distribution function. ' x ' indicate locations of discrete values of \tilde{g} , h^c , and $a(h)$; ' o_i ' indicate locations of discrete values of G , h^b , and $b(h)$.

Following Hibler (1980—Appendix C), a trial version of $a(h)$, termed \hat{a}_l , is formed from (19) by assuming $b(h)$ varies linearly between thickness category boundaries and $g(h)$ is constant within a category, viz

$$\hat{a}_l = \begin{cases} \tilde{g}_l \left[\frac{b_{l+1} + b_l}{2} (h_{l+1}^b - h_l^b) \right] & ; \quad b_{l+1} \geq 0 \\ \tilde{g}_l \left[\frac{b_l}{2} (h^* - h_l^b) \right] & ; \quad b_{l+1} < 0 < b_l \\ 0 & ; \quad b_l < 0 \end{cases} \quad (28)$$

where

$$h^* = h_{l-1}^b - \frac{h_l^b - h_{l-1}^b}{b_l - b_{l-1}} b_{l-1} \quad ; \quad b_l \leq 0 < b_{l-1} \quad (29)$$

The approximate discrete form of the ridge redistribution process, $\hat{\gamma}_{l,m}$, is given by

$$\hat{\gamma}_{l,m} = \begin{cases} \frac{1}{k} & ; \quad h_m^b \leq kh_l^c < h_{m+1}^b \\ & ; \quad m < NL \\ 0 & ; \quad \text{otherwise} \end{cases} \quad (30)$$

$$\hat{\gamma}_{l,NL} = \frac{1}{k} \quad ; \quad kh_l^b > h_{NL-1}^c$$

The notation here is that $\gamma_{l,m} \equiv \gamma(h_l^c, h_m^c)$. The actual discrete redistribution process, $\gamma_{l,m}$, is obtained by normalizing the approximate form (30) so that (23) is satisfied. That is

$$\gamma_{l,m} = \hat{\gamma}_{l,m} \frac{h_l^c}{\sum_{m=1}^{NL} \hat{\gamma}_{l,m} h_m^c} \quad (31)$$

Finally, to satisfy the requirement of area conservation, (14), over a discrete time step, we define $\Delta A = \nabla \cdot \mathbf{u} \Delta t$ as the reduction in area using the definition of $\nabla \cdot \mathbf{u}$ from (16). [Note that ΔA is a dimensionless fractional change in ice-covered area]. This is used to normalize the overall ridge redistribution procedure, viz

$$\tilde{g}_l(t + \Delta t) = \tilde{g}_l(t) - \frac{\Delta A \left[-a_l + \sum_{m=1}^{NL} a_m \gamma_{m,l} \right]}{\sum_{l=1}^{NL} \left[-a_l + \sum_{m=1}^{NL} a_l \gamma_{l,m} \right]} \quad (32)$$

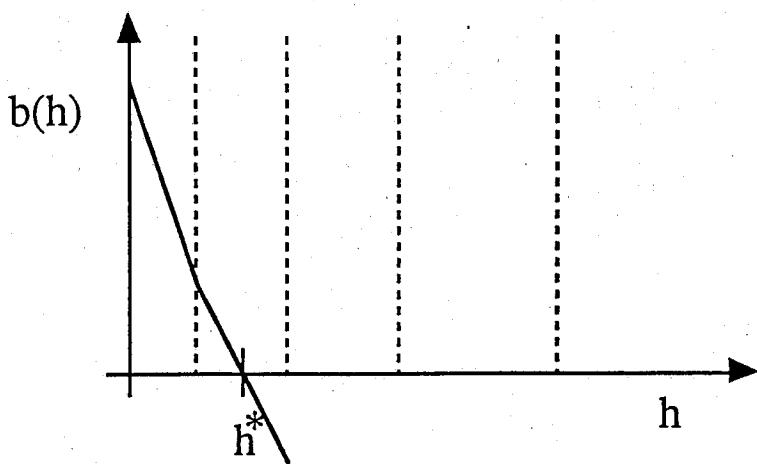
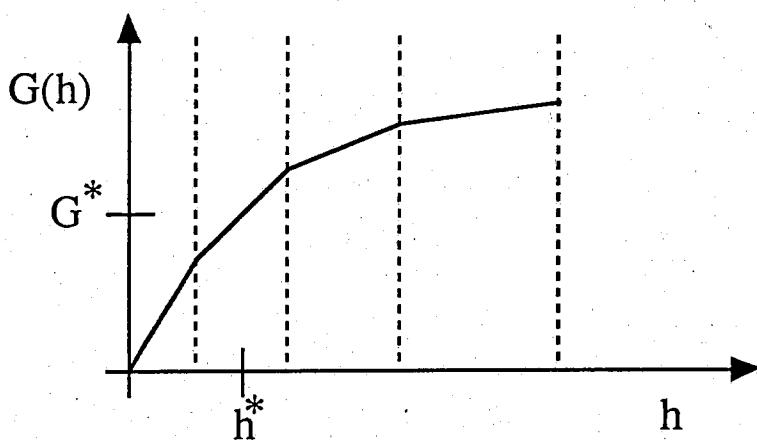
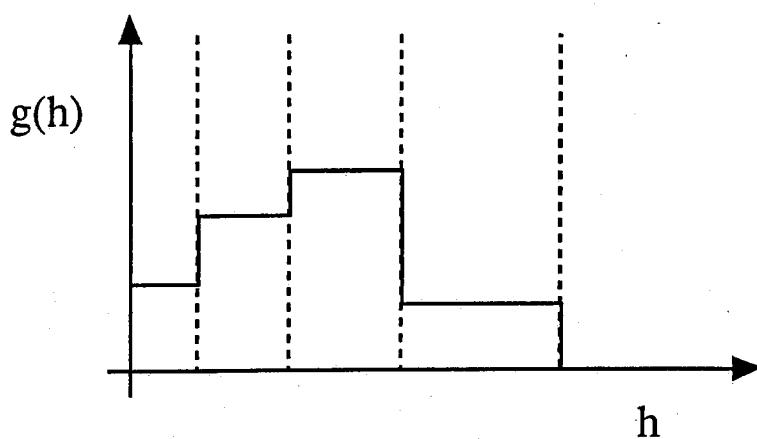


Figure 3. Sketch illustrating construction of the discrete form of $b(h)$, the weighting function for ridge participation.

The second term on the RHS of (32) is effectively the same as in equation (18).

Thermodynamics

The change in $g(h)$ due to thermodynamics is given by the special case of (13),

$$\frac{\partial g}{\partial t} = -\frac{\partial}{\partial h}(fg) \quad (33)$$

This is an advection equation in thickness space with g playing the role of a tracer and f , the ice growth rate with units of m s^{-1} , the role of a velocity. Hibler (1980 — Appendix C) showed that a form of upstream-differencing was required in the discrete case to conserve volume. Also, special consideration had to be given to growth of the thickest ice category and melt of the open water category. The PIC implementation introduces two additional special cases which are described after discussion of the finite-difference approximation to (33).

The growth rate, f , is calculated exactly as in Hibler (1980) — in fact, a slightly modified version of Hibler's heat budget code is used. The growth rates are calculated in each grid cell for each of the NL thickness categories and for open water; these are applied to each particle which resides in that grid cell. The growth rate over open water is defined as

$$f_o = \left. \frac{\partial h}{\partial t} \right|_{h=0} \quad (34)$$

whereas the growth rate of the l^{th} thickness category is

$$f_l = \left. \frac{\partial h}{\partial t} \right|_{h=h_l^b} \quad (35)$$

The finite-difference approximation to (33) is also the same as in (Hibler, 1980).

Referring to Figure 4, the area flux across the l^{th} category boundary is

$$F_l = \begin{cases} 0 & ; \quad l = 1 \text{ or } l = NL + 1 \\ [\max\{0, f_{l-1}\} \tilde{g}_{l-1} + \min\{0, f_l\} \tilde{g}_l] / (h_l^b - h_{l-1}^b) & ; 2 \leq l \leq NL \end{cases} \quad (36)$$

The change in \tilde{g} , due to thermodynamics, over a time step is thus

$$\Delta \tilde{g}_l = -[F_{l+1} - F_l] \Delta t \quad (37)$$

and so

$$\tilde{g}_l(n, t + \Delta t) = \tilde{g}_l(n, t) + \Delta \tilde{g}_l \quad (38)$$

Note that these thermodynamic fluxes of area between ice thickness categories do not change the total ice-covered area.

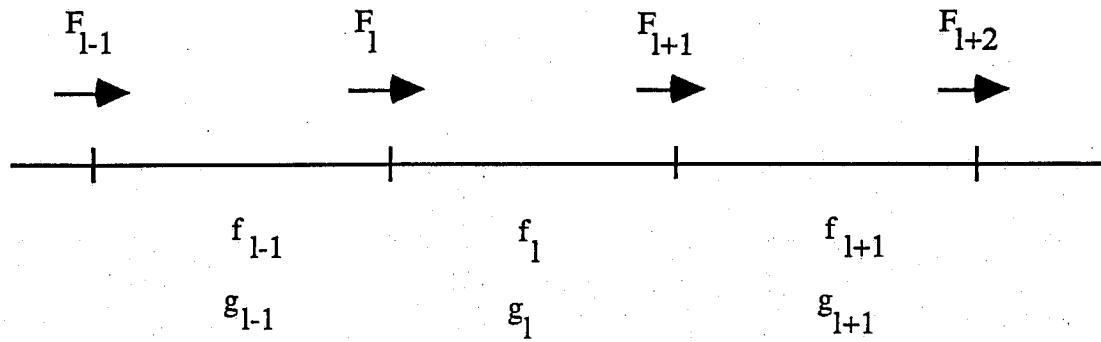


Figure 4. Sketch illustrating ice area flux, F_l , across thickness category boundaries due to thermodynamic growth.

Special Case 1: growth of thickest category

Since there is a finite upper limit on ice thickness, determined by the value of h_{NL+1}^b , growth of ice in the thickest category cannot be represented as a flux of area into a thicker category. Therefore, growth of the thickest ice category is taken to increase the area occupied by this category in such a way as to conserve volume. The change in \tilde{g}_{NL} is taken to be

$$\Delta \tilde{g}_{NL} = \frac{\max\{0, f_{NL}\} \tilde{g}_{NL}}{h_{NL}^c} \Delta t \quad (39)$$

where the numerator is the rate of change of ice volume due to growth of the thickest ice category. Note that this implies an increase in particle area

$$dA(n, t) = \Delta \tilde{g}_{NL} A(n, t) \quad (40)$$

Special Case 2: melt of thinnest category

In the PIC version of the thickness distribution scheme, open water is not contained in the distribution function, therefore melt of the thinnest category of ice results in a decrease of particle area and a change in \tilde{g}_1 calculated, as in (39), from the reduction of ice volume.

$$\Delta \tilde{g}_1 = \frac{\min\{0, f_1\} \tilde{g}_1}{h_1^c} \Delta t \quad (41)$$

The change in particle area associated with this is

$$dA(n,t) = \Delta \tilde{g}_1 A(n,t) \quad (42)$$

Special Case 3: open water growth

As above, since open water is not contained in the thickness distribution function, ice formed over open water, which produces a flux of ice area into the thinnest ice category, must be dealt with as a special case. We define the open water area in a grid cell as

$$A_o(i,j) = \Delta x \Delta y - \sum_{n=1}^N \tilde{S}(i,j,n) A(n,t) \quad (43)$$

where, $\tilde{S}(i,j)$ is an indicator function (Flato, 1993b) which is equal to 1 if particle n is within the i,j grid cell, i.e.

$$\tilde{S}(i,j) = \begin{cases} 1 & \text{if } |X(n,t) - x_{i,j}| \leq \frac{\Delta x}{2} \text{ and } |Y(n,t) - y_{i,j}| \leq \frac{\Delta y}{2} \\ 0 & \text{otherwise} \end{cases} \quad (44)$$

The change in particle area is simply the new ice volume created divided by the centre thickness of the first category,

$$dA(n,t) = \frac{A_o(i,j) \max\{0, f_o\} \Delta t}{N_{i,j} h_1^c} \quad (45)$$

where $N_{i,j}$ is the number of particles in grid cell i,j . The change in \tilde{g}_1 is simply

$$\Delta \tilde{g}_1 = \frac{dA(n,t)}{A(n,t)} \quad (46)$$

The three special cases above all result in changes in ice-covered area and concomitant changes in the relative proportion of area covered by each thickness category, i.e. changes in $g(h)$. After modifying $g(h)$ for each particle as specified by equations (39), (41) and (46), the thickness distribution function must be normalized, viz

$$\tilde{g}_l = \frac{\tilde{g}'_l}{\sum_{k=1}^{NL} \tilde{g}'_k} \quad (47)$$

where the prime indicates the non-normalized value of \tilde{g}_l obtained by adding $\Delta \tilde{g}_l$ from equations (39), (41) and (46).

Special Case 4: open water melt

'Melt' of open water, that is $f_0 < 0$, implies heat absorption by the ocean. McPIC includes a fixed-depth, motionless mixed layer - essentially a heat reservoir - to accomodate this. The amount of heat stored in the mixed layer is denoted $h_{neg}(i,j)$, the equivalent volume per unit area of ice in a given grid cell which could be melted by this stored heat. The heat absorbed by open water is given by

$$dh_{neg}(i,j) = \frac{-\min(0, f_0) A_0(i,j) \Delta t}{\Delta x \Delta y} \quad (48)$$

However, this heat must first be used to melt any available ice (the assumption being that the mixed layer remains at the freezing temperature if any ice is present). For simplicity we will assume that mixed layer heat is used only for lateral melt and that the lateral melt rate is not a function of ice thickness. This implies that $g(h)$ is not affected, only $A(n)$ is reduced. The volume of ice available for melt in a given grid cell is

$$V(i,j) = \sum_{n=1}^N \tilde{S}(i,j,n) \bar{h}(n) A(n) \quad (49)$$

where $\bar{h}(n)$ is the mean thickness of the ice represented by particle n , given by

$$\bar{h}(n) = \int_0^\infty g(h) h dh \quad (50)$$

The maximum volume of ice which can be melted by the mixed layer heat is

$$V_{melt}(i,j) = h_{neg}(i,j) \Delta x \Delta y \quad (51)$$

So, after melting the available ice or using up the available heat, the mixed layer heat storage is

$$h_{neg}(i,j) = -\min\left\{0, \frac{V(i,j) - V_{melt}(i,j)}{\Delta x \Delta y}\right\} \quad (52)$$

and the new ice-covered area in the grid cell is

$$A_{new}(i,j) = \max\left\{0, \frac{(V(i,j) - V_{melt}(i,j)) A_i(i,j)}{V(i,j)}\right\} \quad (53)$$

where $A_i(i,j) = \Delta x \Delta y - A_0(i,j)$ is the ice-covered area in a given grid cell. The change in area in a grid cell is partitioned amongst the particles within the grid cell in proportion to the particle's area, viz

$$A(n,t) = A(n,t) \left[1 + \frac{A_{new}(i,j) - A_i(i,j)}{A_i(i,j)} \right] \quad (54)$$

Discussion of Parameterizations

In developing this model, some physical processes (snowfall for example) were neglected and choices were made regarding parameterizations and numerical methods. Some of these are discussed here. Many of the choices were based on the expectation that this model would be primarily used as an operational forecast model. This implies relatively short (weekly or monthly) model runs in regions near an ice edge. Therefore, processes which have time scales longer than this or which are more important in the central ice pack than in the marginal ice zones may have been idealized somewhat. An effort was made to keep the model as simple as possible initially — more complication can be added later as needed — and to keep the memory and computational requirements as low as possible.

- 1) Perhaps the most obvious difference between the thickness distribution scheme implemented here and those used earlier is the exclusion of open water from the thickness distribution function. The main reason for this was the desire to associate particles with ice: where there is no ice there is no need for particles. If open water (ice of zero thickness) was included in the thickness distribution of a particle, the entire domain would have to be populated with particles thereby increasing the computational cost of the model unnecessarily. So, the philosophy adopted is that particles represent only ice, which moves across a background of open water.
- 2) The internal ice stress is obtained by assuming a Mohr-Coulomb rheology wherein shear strength increases in proportion to normal stress. This assumption is widely-used in granular material studies and was felt to be appropriate for ice edge simulations which are the main focus of the present model. That is, the ice cover in a marginal ice zone is predominantly a collection of individual ice floes which transmit shear stress through inter-particle friction and interlocking. Ice strength was parameterized in terms of the mean thickness and compactness rather than in terms of mechanical energy losses during ridging. In part, this was dictated by the rather crude resolution of the thickness distribution and parameterization of ridge redistribution (discussed below) and the notion that, near the ice edge, the ice strength is expected to be low and so the simple and computationally inexpensive parameterization used here is not expected to be a serious shortcoming. Furthermore, the parameterization of equation (9) has been widely-used in other ice models.

- 3) The ridge redistribution scheme parameterizes the participation of thin ice by allowing only the thinnest 15% of ice-covered area to be available for ridging. This is based partly on field and remote sensing observations which indicate that thin ice is preferentially ridged and, in the presence of thin ice, thick ice is seldom ridged. Aside from these loose statements, this parameterization must be viewed as a guess, albeit a guess which has been used in previous models of this kind (e.g. Flato and Hibler, 1994). A second parameter involved in ridging is the ridge height multiple, k , in equation (24). Although more sophisticated parameterizations of the distribution of ridged ice exist, the distribution of thick ice evolves very slowly and, given the rather crude resolution of the thickness distribution used here, a more complicated scheme did not seem justified. The value of $k=15$ was recommended by Pritchard (1981).
- 4) The simple fixed-depth, motionless, thermodynamic-only mixed-layer was included to allow storage and subsequent release of heat by the ocean. Adding a more complete mixed-layer scheme will likely be the first avenue of development beyond the initial model version described here.

References

- Flato, G.M. 1993a, 'A particle-in-cell sea-ice model', *Atmos.-Ocean*, 3, 339-358.
- Flato, G.M. 1993b, 'A particle-in-cell model for ice edge forecasting', *Proc. OCEANS '93*.
- Flato, G.M. and W.D. Hibler III. 1992, 'Modeling pack ice as a cavitating fluid', *J. Phys. Oceanogr.*, 22, 626-651.
- Flato, G.M. and W.D. Hibler III. 1994, 'Ridging and strength in modelling the thickness distribution of Arctic sea ice.', submitted to *J. Geophys. Res.*
- Harlow, F.H. 1964. The particle-in-cell computing method for fluid dynamics. *Methods Comput. Phys.*, 3, 319-343.
- Hibler, W.D. III. 1980, 'Modeling a variable thickness sea ice cover', *Mon. Wea. Rev.*, 108, 1943-1973.
- Ip, C.F., W.D. Hibler III, and G.M. Flato. 1991. On the effect of rheology on seasonal sea-ice simulations. *Annals of Glaciology*, 15, 17-25.
- Thorndike, A.S. et al. 1975, 'The thickness distribution of sea ice', *J. Geophys. Res.* 80, 4501-4513.
- Pavia, E.G. and B. Cushman-Roisin. 1988. Modeling of oceanic fronts using a particle method. *J. Geophys. Res.*, 93, 3554-3562.
- Pritchard, R.S., 1981, Mechanical behavior of pack ice. in *Mechanics of Structured Media*, A.P.S. Selvadurai (ed.), Elsevier Pub., pp 371-405.

Acknowledgements

This work was funded by the Canadian Panel on Energy Research and Development, PERD #67188. I thank Greg Holloway for many helpful discussions and encouragement, and Bon van Hardenberg for assistance in preparation of this documentation.

Appendix 1 — McPIC Code Listing

The Fortran source code in this section includes a complete example: no external data files are required since the forcing is specified for this test case. The comments in the code clearly indicate where the code can be adapted for user supplied field data or for modification of the thickness boundaries, ocean currents or oceanic heat flux, or other parameters of specific interest. More specific details about the code, broken down by line numbers and including cross referencing to equation numbers in the theory overview, have been described in a previous section.

The code is arranged in sections which:

- specify the physical parameters,
- initialize and specify boundaries,
- load the time invariant input (mean currents, fixed flux etc.)
- go through the time steps, in which
 - new data are read,
 - subroutines called to compute changes,
 - arrays are updated and
 - results are written files at end of day (or run)
- all subroutines are then listed in alphabetical order.

```
1 c
2 program mcpic
3 c
4 c =====
5 c
6 c          McPIC (version 1.0)
7 c
8 c          Multi-category Particle-In-Cell Sea Ice Model.
9 c          by
10 c          Gregory M. Flato
11 c          Institute of Ocean Sciences, Sidney, BC.
12 c
13 c This model is a research tool in the early stages of development.
14 c It is available for use by others, but those wishing to do so should
15 c check the code carefully and be sure that the assumptions and
16 c parameter choices are appropriate for their particular application.
17 c
18 c =====
19 c
20 c Test case:      vortex wind field centered at the ice edge, and initial
21 c      ice at 9 particles per cell, with thickness distribution over 6
22 c      levels as found on Labrador Shelf in March, evenly distributed
23 c      over all cells in a rectangle in the upper 2/3 of the left half
24 c      of the model domain, and with oceanic heat flux ranging from
25 c      -150 at upper left to -250 at lower right of the domain.
26 c
27 c Released with updated test case routines and documentation:
28 c 25 March 1994 - Bon van Hardenberg - I.O.S.
29 c
30 c =====
31 c
32 c Model based on two-level version described in:
33 c
34 c Flato, G.M., "A Particle-in-Cell Sea-Ice Model", Atmos.-
35 c          Ocean, 31(3), 339-358, 1993.
36 c Flato, G.M., "A Particle-in-Cell Model for Ice Edge Forecasting",
37 c          Proc. OCEANS '93.
38 c Overview:
39 c
40 c -dynamics based on Mohr-Coulomb extension of cavitating fluid
41 c sea ice model, Flato and Hibler (J.Phys.Oceanogr.,22,626-651,
42 c 1992).
43 c
44 c -particle advection and interpolation similar to PIC ocean model
45 c of Pavia and Cushman-Roisin (J.Geophys.Res.,93,3554-3562,1988).
46 c
47 c -Thermodynamics essentially as described in W.D. Hibler III
48 c (Mon.Wea.Rev.,108,1943-1973,1980, Appendix B).
49 c
50 c -Thickness distribution based on theory of Thorndike et al.,
51 c (J.Geophys.Res.,80,4501-4513,1975).
```

```

52 c
53 c Revision history:
54 c
55 c     Initial version: 8/October/93
56 c     Corrected thermodynamic terms 9/Dec/93
57 c     Updates test case routines 15/Mar/94
58 c
59 c Some Definitions:
60 c
61 c     rectangular grid (NX+1 by NY+1 grid cells)
62 c     NL is the number of thickness levels
63 c     part(n,k) contains particle information, particle number 'k'
64 c         n=1 -> X location (in grid coordinates)
65 c         n=2 -> Y location (in grid coordinates)
66 c         n=3 -> Particle Volume (units are km**3)
67 c         n=4 -> Particle Area (units are 10^6 m**2)
68 c         n>4 -> Area fraction of various thickness categories
69 c     uice(i,j) & vice(i,j) are ice velocity components at grid nodes
70 c     heff(i,j) is mean thickness (volume per unit area)
71 c     area(i,j) is compactness (ice-covered area per unit area)
72 c
73 c =====
74 c
75 implicit real*8(a-h,o-z)
76 parameter (nx=30,ny=60)
77 parameter (kmax=17500, nl=6, ndes=nl+4)
78 common/one/XX(nx,ny),YY(nx,ny),Ax(nx,ny),Ay(nx,ny),
79 <Bx(nx,ny),By(nx,ny),uice(nx,ny,3),vice(nx,ny,3),
80 <heff(nx+1,ny+1,3),area(nx+1,ny+1,3),presur(nx+1,ny+1),
81 <xpress(nx+1,ny+1),heffm(nx+1,ny+1),uvmx(nx,ny),
82 <uvmy(nx,ny),uvmb(nx,ny)
83 common/two/deltax,deltay,deltat
84 common/three/dwatn(nx,ny),dairn(nx,ny),gwatx(nx,ny),
85 <gwaty(nx,ny),gairx(nx,ny),gairy(nx,ny),wta,ata,fcor,rhoice
86 common/four/tair(nx+1,ny+1),qa(nx+1,ny+1),
87 <flo(nx+1,ny+1),fsh(nx+1,ny+1),ug(nx+1,ny+1)
88 common/six/tmix(nx+1,ny+1),tice(nx+1,ny+1),hneg(nx+1,ny+1),
89 <fw(nx+1,ny+1)
90 common/pic/part(ndes,kmax),partO(2,kmax),aunit,hunit,
91 <hb(nl+1),hc(nl),knum
92 common/redist/rk,Gstar
93 dimension heffa(nx+1,ny+1),uicea(nx,ny),vicea(nx,ny),
94 <areaa(nx+1,ny+1)
95 c
96 c open files for forcing and simulation results
97 c
98 c*** TEST CASE: NO SEPARATE FORCING FILES
99 c (code documents how test case forcing is specified)
100 c
101 open(10,file='area.out',form='formatted')
102 open(11,file='thick.out',form='formatted')
103 open(12,file='velocity.out',form='formatted')

```

```

104      open(13,file='parx.out',form='formatted')
105      open(14,file='pary.out',form='formatted')
106      open(15,file='parv.out',form='formatted')
107      open(16,file='area1.out',form='formatted')
108      open(17,file='area2.out',form='formatted')
109      open(18,file='area3.out',form='formatted')
110      open(19,file='area4.out',form='formatted')
111      open(20,file='area5.out',form='formatted')
112      open(21,file='area6.out',form='formatted')

113 c
114 c user-specified parameters
115 c
116      nday=10          !number of days in simulation
117      ntpd=10          !number of time steps per day
118      deltax=20000.D+00 !lx-grid size (m)
119      deltay=20000.D+00 !ly-grid size (m)
120      fcor=1.46D-04    !Coriolis parameter (s^-1)
121      rhoice=0.92*1.D+03 !ice density (kg m^-3)
122      pi=acos(-1.D+00) !pi
123      deg=pi/180.       !conversion from degrees to radians
124      ata=25.*deg      !air drag turning angle (radians)
125      wta=25.*deg      !water drag turning angle (radians)
126      rhoCa=1.56D-03   !air density times drag coefficient (kg m^-3)
127      rhoCw=5.5D+00     !water density times drag coefficient (kg m^-3)
128      pstar=27.5D+03   !compressive strength parameter (Pa)
129      phi=30.D+00       !shear strength parameter (degrees)
130      aunit=1.D+6       !conversion factor for area units
131      hunit=1.D+9       !conversion factor for volume units
132      kadd=3            !number of particles to add if none exist
133      rk=15.D+00        !ridge thickness parameter
134      Gstar=0.15D+00    !ridge participation parameter
135      hb(1)=0.D+00      !thickness category boundaries
136      hb(2)=0.15D+00
137      hb(3)=0.30D+00
138      hb(4)=1.00D+00
139      hb(5)=2.50D+00
140      hb(6)=5.00D+00
141      hb(7)=10.00D+00

142 c
143 c derived constants
144 c
145      deltat=86400./float(ntpd) !time step length in seconds
146      nts=nday*ntpdi           !number of time steps in simulation
147      cosata=cos(ata)
148      sinata=sin(ata)
149      coswta=cos(wta)
150      sinwta=sin(wta)
151      do l=1,nl
152          hc(l)=(hb(l)+hb(l+1))/2. !thickness category centres
153      enddo
154 c
155 c set up boundary masks (velocity masks from heffm() )

```

```

156 c
157   call bnd
158 c
159 c initialize some arrays
160 c
161   do j=1,ny
162     do i=1,nx
163       uicea(i,j)=0.
164       vicea(i,j)=0.
165       gwatx(i,j)=0.
166       gwaty(i,j)=0.
167       do k=1,3
168         uice(i,j,k)=0.
169         vice(i,j,k)=0.
170       enddo
171     enddo
172   enddo
173   do j=1,ny+1
174     do i=1,nx+1
175       hneg(i,j)=0.
176       tice(i,j)=273.0D+00
177       presur(i,j)=0.
178       do k=1,3
179         heff(i,j,k)=0.0
180         area(i,j,k)=0.0
181       enddo
182     enddo
183   enddo
184 c
185 c initialize particle location and thickness distribution
186 c (e.g. thickness distribution might come from satellite data).
187 c -for this example calculation, particles are regularly
188 c spaced with 9 particles per grid cell and the thickness
189 c distribution is taken everywhere to represent roughly
190 c the March thickness distribution off the Newfoundland
191 c Coast observed by Symonds (Proc. E.Coast Workshop on Sea
192 c Ice, BIO, 1986).
193 c
194   at=1.0          !total ice-covered area fraction
195
196   g1=0.31          ! sum(gi)=1
197   g2=0.25
198   g3=0.42
199   g4=0.02
200   g5=0.00
201   g6=0.00
202 c
203   k=0
204   do k1=1,123
205     do k2=1,45
206       k=k+1
207       part(1,k)=2.5+(1./6.)+float(k2-1)/3.

```

```

208      part(2,k)=18.5+(1./6.)+float(k1-1)/3.
209      part(4,k)=at*0.1111*deltax*deltay/aunit
210      part(5,k)=g1
211      part(6,k)=g2
212      part(7,k)=g3
213      part(8,k)=g4
214      part(9,k)=g5
215      part(10,k)=g6
216      enddo
217      enddo
218      knum=k
219 c
220 c interpolate thickness and compactness
221 c
222     call interp4
223
224     print*,heff(10,30,1),area(10,30,1)',heff(10,30,1),area(10,30,1)
225
226 c
227 c read in annual average ocean current field
228 c*** TEST CASE: no ocean currents: gwatx(i,j) = gwaty(i,j) = 0
229 c
230 c read in annual average oceanic heat flux
231 c*** TEST CASE: specified heat flux field based loosely on Symonds (1986)
232 c oceanic heat flux increases from upper left to lower right corner:
233 c
234     do j=1,ny+1
235       do i=1,nx+1
236         XD=sqrt(float(i-1)**2+float(61-j)**2)
237         XB=sqrt(float(31-i)**2+float(j-1)**2)
238         XL=XB+XD
239         fw(i,j)=-150.-(XD/XL)*100.
240       enddo
241     enddo
242 c
243 c ****
244 c begin time-stepping
245 c ****
246 c
247 c do for number of time steps in simulation
248 c
249   do 1000 iday=1,nts
250   print,''
251   print,'* time step ',iday
252 c
253 c calculate strength
254 c
255   do j=1,ny+1
256     do i=1,nx+1
257 c       xmpress(i,j)=0. !strength = 0 for free drift test
258       xmpress(i,j)=pstar*heff(i,j,1)*exp(-20.*(1.-area(i,j,1)))
259     enddo

```

```

260      enddo
261  c
262  c read wind field
263  c
264  c*** TEST CASE: the following lines specify a constant vortex wind field
265  c   centered at x=20.1, y=30.1 with max 30m/s at 30km from centre
266  c
267  xlam=18.D+05
268  omega=0.5D-03
269  xc=20.1
270  yc=30.1
271  do j=1,ny
272    do i=1,nx
273      xu=float(i)+0.5-xc
274      yu=float(j)+0.5-yc
275      xu=xu*deltax
276      yu=yu*deltay
277      theta=atan2(yu,xu)
278      R=sqrt(xu**2+yu**2)
279      u11=-(xlam/R)*sin(theta)
280      u22=-(omega*R)*sin(theta)
281      xv=float(i)+1.-xc
282      yv=float(j)+0.5-yc
283      xv=xv*deltax
284      yv=yv*deltay
285      theta=atan2(yv,xv)
286      R=sqrt(xv**2+yv**2)
287      v11=(xlam/R)*cos(theta)
288      v22=(omega*R)*cos(theta)
289      u=min(abs(u11),abs(u22))*sign(1.D+00,u11)
290      v=min(abs(v11),abs(v22))*sign(1.D+00,v11)
291      gairx(i,j)=u
292      gairy(i,j)=v
293    enddo
294  enddo
295  c
296  c put minimum value on wind velocity components to avoid
297  c problems with inverse trig function in FORCES
298  c
299    do j=1,ny
300      do i=1,nx
301        if(abs(gairx(i,j)).lt.0.00005) gairx(i,j)=0.00005
302        if(abs(gairy(i,j)).lt.0.00005) gairy(i,j)=0.00005
303      enddo
304    enddo
305  c
306  c... Modified Euler time step for non-linear drag terms
307  c
308    do kme=1,2
309  c
310  c average velocities here for second iteration
311  c

```

```

312 if(kme.ge.2) then
313   do j=1,ny
314     do i=1,nx
315       uice(i,j,1)=0.5*(uice(i,j,1)+uice(i,j,3))
316       vice(i,j,1)=0.5*(vice(i,j,1)+vice(i,j,3))
317     enddo
318   enddo
319 endif
320 c
321 c update velocity array
322 c
323   do j=1,ny
324     do i=1,nx
325       uice(i,j,3)=uice(i,j,1)
326       vice(i,j,3)=vice(i,j,1)
327     enddo
328   enddo
329 c
330 c calculate drag coefficients
331 c
332   do j=1,ny
333     do i=1,nx
334       dairn(i,j)=rhoCa*sqrt(gairx(i,j)**2+gairy(i,j)**2)
335       dwatn(i,j)=rhoCw*sqrt((uice(i,j,1)-gwatx(i,j))**2
336       < +(vice(i,j,1)-gwaty(i,j))**2)
337 c
338 c put minimum on drag coefficient to avoid spuriously large velocities
339 c when near zero drag occurs.
340 c
341   dwatn(i,j)=max(dwatn(i,j),0.25D+00)
342 c
343 c*** Option: linear drag coefficients ***
344 c
345 c Ref: Flato, G.M. and W.D. Hibler III, "On a simple sea-ice dynamics
346 c model for climate studies", Annals of Glaciology, 14, 72-77, 1990.
347 c NOTE: for linear drag calculation, need only one loop over this time-
348 c stepping scheme; i.e. replace statement ~25 lines up by "do kme=1,1"
349 c
350 c   dairn(i,j)=0.01256
351 c   dwatn(i,j)=0.6524
352 c
353 c*** End Option ***
354 c
355   dwatx=dwatn(i,j)
356   dwaty=dwatn(i,j)
357   Ax(i,j)=dwatx*coswta
358   Bx(i,j)=((heff(i+1,j+1,1)+heff(i,j+1,1))/2.*rhoice*fcor)
359   < +(dwatx*sinwta)
360   Ay(i,j)=dwaty*coswta
361   By(i,j)=((heff(i+1,j+1,1)+heff(i+1,j,1))/2.*rhoice*fcor)
362   < +(dwaty*sinwta)
363 c

```

```

364      enddo
365      enddo
366 c
367 c calculate forcing fields
368 c
369     if(kme.eq.1) call force
370 c
371 c calculate free drift velocity field
372 c
373     call freed
374 c
375 c calculate corrected velocity and pressure fields
376 c
377     call nlcav
378 c
379 c... end of Modified Euler time step
380 c
381     enddo
382 c
383 c apply shear strength using Mohr-Coulomb rheology
384 c
385     call shear(phi)
386 c
387 c advect particles
388 c
389 c...first step in predictor/corrector
390     do k=1,knum
391         part0(1,k)=part(1,k)
392         part0(2,k)=part(2,k)
393     enddo
394     deltt=deltat
395     deltat=deltat/2.
396     call picadvect
397 c...second step in predictor/corrector
398     deltat=deltt
399     call picadvect
400 c
401 c read in thermodynamic fields
402 c
403 c***thermodynamic parameters for test case
404 c after Symonds (1986) - March, April & May
405 c
406     do J=1,NY+1
407     do I=1,NX+1
408 c atmospheric quantities assumed to decrease linearly from south to north
409 c March
410 c     TAIR(I,J)=270.5D+00-(8./60.)*float(j-1)
411 c     FLO(I,J)=220.-(36./60.)*float(j-1)
412 c     FSH(I,J)=104.-(10./60.)*float(j-1)
413 c April
414     TAIR(I,J)=274.D+00-(6./60.)*float(j-1)
415     FLO(I,J)=233.-(28./60.)*float(j-1)

```

```

416     FSH(I,J)=151.-(7./60.)*float(j-1)
417 c May
418 c   TAIR(I,J)=278.5D+00-(4./60.)*float(j-1)
419 c   FLO(I,J)=250.-(20./60.)*float(j-1)
420 c   FSH(I,J)=195.-(4./60.)*float(j-1)
421 c
422     QA(I,J)=3.0D-03 ! 80% rel. hum. at 0 degrees C.
423   enddo
424   enddo
425 c
426 c do thermodynamic growth/decay
427 c (Note: thermodynamics done before deformation so that area change
428 c due to growth/melt is applied before constraint that A .le. 1)
429 c
430   call GROWTH(kadd)
431 c
432   print*, '# of Particles is now', knum
433 c
434 c calculate change in particle area and ridge redistribution
435 c as a result of convergent deformation
436 c
437   call PICDEFORM
438 c
439 c update mean thickness and compactness arrays
440 c
441   call INTERP4
442 c
443 c write out total ice volume and mixed-layer heat storage
444 c
445   vtot=0.D+00
446   htot=0.D+00
447   do k=1,knum
448     vtot=vtot+part(3,k)
449   enddo
450   do j=1,ny+1
451     do i=1,nx+1
452       htot=htot+hneg(i,j)*heffm(i,j)*deltax*deltay/hunit
453     enddo
454   enddo
455
456   print*, ' total ice volume (km**3) =', vtot
457   print*, ' total ML heat storage (km**3) =', htot
458 c
459 c interpolate velocity field back to B-Grid for plotting
460 c
461   do j=1,ny
462     do i=1,nx
463       uice(i,j,2)=uice(i,j,1)
464       vice(i,j,2)=vice(i,j,1)
465     enddo
466   enddo
467   do j=2,ny

```

```

468      do i=2,nx
469          uicea(i,j)=uicea(i,j)+uvmb(i,j)*0.5*(uice(i,j,2)
470          <           +uice(i,j-1,2))
471          vicea(i,j)=vicea(i,j)+uvmb(i,j)*0.5*(vice(i,j,2)
472          <           +vice(i-1,j,2))
473      enddo
474      enddo
475 c
476      do j=1,ny+1
477          do i=1,nx+1
478              heffa(i,j)=heff(i,j,1)
479              areaa(i,j)=area(i,j,1)
480              if(heffm(i,j).lt.0.1) heffa(i,j)=-1.
481              if(heffm(i,j).lt.0.1) areaa(i,j)=-1.
482          enddo
483      enddo
484 c
485 c print out area, thickness and velocity fields at end of each day
486 c
487     rmo=1.
488     if(mod(iday,ntpd).eq.0) then
489         rmo=float(ntpd)
490 c
491 c... or instead, printout only at end of run:
492 c... if(mod(iday,nts).eq.0) then
493 c...   rmo=float(nts)
494 c
495     write(10,102) ((areaa(i,j),i=1,nx+1),j=1,ny+1) ! 'area.out'
496 c
497     write(11,102) ((heffa(i,j),i=1,nx+1),j=1,ny+1) ! 'thick.out'
498 c
499     write(12,103) ((uicea(i,j)/rmo,vicea(i,j)/rmo,
500     <           i=1,nx),j=1,ny) ! 'velocity.out'
501 c
502 c print out particle position and area
503 c
504     do k=1,knum
505         write(13,104) part(1,k) !'parx.out'
506         write(14,104) part(2,k) !'pary.out'
507         write(15,104) part(4,k) !'parv.out'
508     enddo
509 c
510 c print out area concentration of each thickness category
511 c
512 do n=5,10
513 call INTERPC(areaa,n)
514 do j=1,ny+1
515 do i=1,nx+1
516 if (heffm(i,j).lt.0.1) areaa(i,j)=-1.
517 enddo
518 enddo
519 write(11+n,102) ((areaa(i,j),i=1,nx+1),j=1,ny+1)

```

```

520    enddo
521  c
522  c and initialize for next printout
523  c
524      do j=1,ny
525      do i=1,nx
526          uicea(i,j)=0.
527          vicea(i,j)=0.
528      enddo
529      enddo
530  c
531      endif
532  c
533  1000 continue
534  c
535  c ****
536  c end time-stepping
537  c ****
538
539  100 format(12f3.0)
540  101 format(///)
541  102 format(12g10.4)
542  103 format(2g16.5)
543  104 format(g16.5)
544  105 format(10f7.3)
545  c
546      stop
547      end
548  c
549  c ===== SUBROUTINES =====
550  c
551  c -----
552      subroutine BND
553  c -----
554  c generates velocity boundary masks based on thickness mask specified
555  c in block data "comint"
556  c
557      implicit real*8(a-h,o-z)
558      parameter (nx=30,ny=60)
559      common/one/XX(nx,ny),YY(nx,ny),Al(nx,ny),Ap(nx,ny),
560      <Bl(nx,ny),Bp(nx,ny),uice(nx,ny,3),vice(nx,ny,3),
561      <heff(nx+1,ny+1,3),area(nx+1,ny+1,3),presur(nx+1,ny+1),
562      <xmpress(nx+1,ny+1),heffm(nx+1,ny+1),uvmx(nx,ny),
563      <uvmy(nx,ny),uvmb(nx,ny)
564  c
565      do j=1,ny
566      do i=1,nx
567          uvmx(i,j)=0.
568          uvmy(i,j)=0.
569          uvmb(i,j)=0.
570      enddo
571      enddo

```

```

572 c
573 c calculate velocity mask in B-Grid and C-Grid
574 c
575 do j=1,ny
576 do i=1,nx
577   htx=heffm(i,j+1)+heffm(i+1,j+1)
578   hty=heffm(i+1,j)+heffm(i+1,j+1)
579   htb=heffm(i,j+1)+heffm(i+1,j+1)+heffm(i,j)+heffm(i+1,j)
580   if(htx.gt.1.5) uvmx(i,j)=1.0
581   if(hty.gt.1.5) uvmy(i,j)=1.0
582   if(htb.gt.3.5) uvmb(i,j)=1.0
583 enddo
584 enddo
585 c
586 return
587 end
588 c
589 c -----
590 SUBROUTINE BUDGET(HICE1,FICE,KOPEN)
591 c -----
592 c calculates surface energy budget and hence ice growth or melt rate.
593 c modified from a subroutine of W.D. Hibler III, based on Heat Budget
594 c code described in Mon. Wea. Rev., 108, 1943-1973, 1980, Appendix B.
595 c
596 c
597 implicit real*8(a-h,o-z)
598 parameter (nx=30,ny=60)
599 common/one/XX(nx,ny),YY(nx,ny),Ax(nx,ny),Ay(nx,ny),
600 <Bx(nx,ny),By(nx,ny),uice(nx,ny,3),vice(nx,ny,3),
601 <heff(nx+1,ny+1,3),area(nx+1,ny+1,3),presur(nx+1,ny+1),
602 <xmpress(nx+1,ny+1),heffm(nx+1,ny+1),uvmx(nx,ny),
603 <uvmy(nx,ny),uvmb(nx,ny)
604 common/two/deltax,deltay,deltat
605 common/three/dwatn(nx,ny),dairn(nx,ny),gwatx(nx,ny),
606 <gwaty(nx,ny),gairx(nx,ny),gairy(nx,ny),wta,ata,fcor,rhoice
607 common/four/tair(nx+1,ny+1),qa(nx+1,ny+1),
608 <flo(nx+1,ny+1),fsh(nx+1,ny+1),ug(nx+1,ny+1)
609 common/six/tmix(nx+1,ny+1),tice(nx+1,ny+1),hneg(nx+1,ny+1),
610 <fw(nx+1,ny+1)
611 DIMENSION HICE(NX+1,NY+1),FICE(NX+1,NY+1),A1(NX+1,NY+1),
612 <A2(NX+1,NY+1),A3(NX+1,NY+1),B(NX+1,NY+1),ALB(NX+1,NY+1),
613 <HICE1(NX+1,NY+1)
614 c
615 QS1=0.622D+00/1013.0D+00
616 C1=2.7798202D-06
617 C2=-2.6913393D-03
618 C3=0.97920849D+00
619 C4=-158.63779D+00
620 C5=9653.1925D+00
621 Q0=1.0D-06/302.0D+00
622 TB=271.2D+00
623 IMAX=5

```

```

624      D1=2.284D+00
625      D1W=5.6875D+03
626      D1I=6.4474D+03
627      D3=5.5D-08
628      TMELT=273.16D+00
629      TMELTP=273.159D+00
630 c   IF(KOPEN.GT.0) GO TO 51
631      DO 110 J=1,NY+1
632      DO 110 I=1,NX+1
633      HICE(I,J)=1.0D+00
634
635 110  CONTINUE
636      DO 101 J=1,NY+1
637      DO 101 I=1,NX+1
638      ALB(I,J)=0.1E+00
639      A1(I,J)=0.90E+00*FSH(I,J)+FLO(I,J)+D1*UG(I,J)*TAIR(I,J)
640      ?+D1W*UG(I,J)*QA(I,J)
641      B(I,J)=QS1*6.11E+00*EXP(17.2694E+00*(TMIX(I,J)-TMELT)
642      ?/(TMIX(I,J)-TMELT+237.3E+00))
643      A2(I,J)=-D1*UG(I,J)*TMIX(I,J)-D1W*UG(I,J)*B(I,J)
644      ?-D3*(TMIX(I,J)**4)
645 101  CONTINUE
646      GO TO 52
647 51   CONTINUE
648      ITER=0
649      DO 102 J=1,NY+1
650      DO 102 I=1,NX+1
651      HICE(I,J)=MAX(HICE1(I,J),0.05D+00)
652 102  CONTINUE
653      DO 103 J=1,NY+1
654      DO 103 I=1,NX+1
655      ALB(I,J)=0.75E+00
656      IF(TICE(I,J).GT.TMELTP) ALB(I,J)=0.66D+00
657 103  CONTINUE
658      DO 104 J=1,NY+1
659      DO 104 I=1,NX+1
660      A1(I,J)=(1.0-ALB(I,J))*FSH(I,J)+FLO(I,J)+D1*UG(I,J)*TAIR(I,J)
661      ?+D1*UG(I,J)*QA(I,J)
662 104  CONTINUE
663 60   CONTINUE
664      DO 105 J=1,NY+1
665      DO 105 I=1,NX+1
666      B(I,J)=QS1*6.11*EXP(21.8746*(TICE(I,J)-TMELT)
667      1/(TICE(I,J)-TMELT+265.5))
668      A3(I,J)=D1I*UG(I,J)*B(I,J)*21.8746*265.5/((TICE(I,J)
669      1-TMELT+265.5)**2)
670      A2(I,J)=-D1*UG(I,J)*TICE(I,J)-D1I*UG(I,J)*B(I,J)
671      ?-D3*(TICE(I,J)**4)
672      OC1=-D1*UG(I,J)*TICE(I,J)
673      OC2=-D1I*UG(I,J)*B(I,J)
674      OC3=-D3*(TICE(I,J)**4)
675      B(I,J)=(2.1656E+00/HICE(I,J))

```

```

676     A3(I,J)=A3(I,J)+4.0E+00*D3*(TICE(I,J)**3)+B(I,J)
677     ?+D1*UG(I,J)
678     B(I,J)=B(I,J)*(TB-TICE(I,J))
679 105 CONTINUE
680     IF(ITER.GE.IMAX) GO TO 52
681     DO 106 J=1,NY+1
682     DO 106 I=1,NX+1
683     TICE(I,J)=TICE(I,J)+(A1(I,J)+A2(I,J)+B(I,J))/A3(I,J)
684 106 CONTINUE
685     ITER=ITER+1
686     IF(ITER.LT.IMAX) GO TO 60
687     ITER=IMAX+1
688     DO 107 J=1,NY+1
689     DO 107 I=1,NX+1
690     TICE(I,J)=MIN(TICE(I,J),TMELT)
691 107 CONTINUE
692     GO TO 60
693 52 CONTINUE
694     DO 108 J=1,NY+1
695     DO 108 I=1,NX+1
696     FICE(I,J)=Q0*(FW(I,J)-A1(I,J)-A2(I,J))
697 108 CONTINUE
698     RETURN
699     END
700 c
701 c -----
702 BLOCK DATA COMINT
703 c -----
704 c subroutine to specify thickness mask in COMMON block.
705 c
706 c
707     implicit real*8(a-h,o-z)
708     parameter (nx=30,ny=60)
709     parameter (n1=ny+1)
710     common/one/XX(nx,ny),YY(nx,ny),Al(nx,ny),Ap(nx,ny),
711     <Bl(nx,ny),Bp(nx,ny),uice(nx,ny,3),vice(nx,ny,3),
712     <heff(nx+1,ny+1,3),area(nx+1,ny+1,3),presur(nx+1,ny+1),
713     <xpress(nx+1,ny+1),heffm(nx+1,ny+1),uvmx(nx,ny),
714     <uvmy(nx,ny),uvmb(nx,ny)
715 c
716 c specify thickness mask
717 c
718     data(heffm(1,j),j=1,n1)/61*0./
719     data(heffm(2,j),j=1,n1)/61*0./
720     data(heffm(3,j),j=1,n1)/2*0.,57*1.,2*0./
721     data(heffm(4,j),j=1,n1)/2*0.,57*1.,2*0./
722     data(heffm(5,j),j=1,n1)/2*0.,57*1.,2*0./
723     data(heffm(6,j),j=1,n1)/2*0.,57*1.,2*0./
724     data(heffm(7,j),j=1,n1)/2*0.,57*1.,2*0./
725     data(heffm(8,j),j=1,n1)/2*0.,57*1.,2*0./
726     data(heffm(9,j),j=1,n1)/2*0.,57*1.,2*0./
727     data(heffm(10,j),j=1,n1)/2*0.,57*1.,2*0./

```

```

728 data(heffm(11,j),j=1,n1)/2*0.,57*1.,2*0./
729 data(heffm(12,j),j=1,n1)/2*0.,57*1.,2*0./
730 data(heffm(13,j),j=1,n1)/2*0.,57*1.,2*0./
731 data(heffm(14,j),j=1,n1)/2*0.,57*1.,2*0./
732 data(heffm(15,j),j=1,n1)/2*0.,57*1.,2*0./
733 data(heffm(16,j),j=1,n1)/2*0.,57*1.,2*0./
734 data(heffm(17,j),j=1,n1)/2*0.,57*1.,2*0./
735 data(heffm(18,j),j=1,n1)/2*0.,57*1.,2*0./
736 data(heffm(19,j),j=1,n1)/2*0.,57*1.,2*0./
737 data(heffm(20,j),j=1,n1)/2*0.,57*1.,2*0./
738 data(heffm(21,j),j=1,n1)/2*0.,57*1.,2*0./
739 data(heffm(22,j),j=1,n1)/2*0.,57*1.,2*0./
740 data(heffm(23,j),j=1,n1)/2*0.,57*1.,2*0./
741 data(heffm(24,j),j=1,n1)/2*0.,57*1.,2*0./
742 data(heffm(25,j),j=1,n1)/2*0.,57*1.,2*0./
743 data(heffm(26,j),j=1,n1)/2*0.,57*1.,2*0./
744 data(heffm(27,j),j=1,n1)/2*0.,57*1.,2*0./
745 data(heffm(28,j),j=1,n1)/2*0.,57*1.,2*0./
746 data(heffm(29,j),j=1,n1)/2*0.,57*1.,2*0./
747 data(heffm(30,j),j=1,n1)/61*0./
748 data(heffm(31,j),j=1,n1)/61*0./
749 end
750 c
751 c -----
752 subroutine FORCE
753 c -----
754 c calculates velocity-independent forcing terms in B-grid.
755 c As a by-product, this subroutine calculates the free drift velocity
756 c field in the B-grid which is used as an initial guess to speed up
757 c C-grid free drift iteration in subroutine FREED.
758 c
759 c
760 implicit real*8(a-h,o-z)
761 parameter (nx=30,ny=60)
762 common/one/XX(nx,ny),YY(nx,ny),Al(nx,ny),Ap(nx,ny),
763 <Bl(nx,ny),Bp(nx,ny),uice(nx,ny,3),vice(nx,ny,3),
764 <heff(nx+1,ny+1,3),area(nx+1,ny+1,3),presur(nx+1,ny+1),
765 <xmpress(nx+1,ny+1),heffm(nx+1,ny+1),uvmx(nx,ny),
766 <uvmy(nx,ny),uvmb(nx,ny)
767 common/two/deltax,deltay,deltat
768 common/three/dwatn(nx,ny),dairn(nx,ny),gwatx(nx,ny),
769 <gwaty(nx,ny),gairx(nx,ny),gairy(nx,ny),wta,ata,fcor,rhoice
770 c
771 c specify some constants
772 c
773 cosata=cos(ata)
774 sinata=sin(ata)
775 coswta=cos(wta)
776 sinwta=sin(wta)
777 sintmp=sin(wta-ata)
778 costmp=cos(wta-ata)
779 c

```

```

780 c calculate linear free drift
781 c
782   do j=1,ny
783   do i=1,nx
784     heffa=(heff(i,j,1)+heff(i+1,j,1)+heff(i,j+1,1)
785     < +heff(i+1,j+1,1))/4.
786     delta=57.29575*atan((sintmp*dwatn(i,j)+rhoice*heffa*fcor
787     < *cosata)/(costmp*dwatn(i,j)+rhoice*heffa*fcor*sinata))
788     Ug=sqrt(gairx(i,j)**2+gairy(i,j)**2)
789     Ui=(dairn(i,j)*Ug*cos(delta*.0174533))/(
790     < (costmp*dwatn(i,j)+rhoice*heffa*fcor*sinata)
791     gamma=57.29575*atan2(gairy(i,j),gairx(i,j))
792     uice(i,j,1)=(Ui*cos((gamma-delta)*.0174533)+gwatx(i,j))
793     vice(i,j,1)=(Ui*sin((gamma-delta)*.0174533)+gwayt(i,j))
794   enddo
795   enddo
796 c
797 c average forces to C-Grid
798 c
799   do j=2,ny-1
800   do i=2,nx-1
801     Aij=dwatn(i,j)*coswta
802     Bij=((heff(i+1,j+1,1)+heff(i,j+1,1)+heff(i,j,1)
803     < +heff(i+1,j,1))/4.*rhoice*fcor)+(dwatn(i,j)*sinwta)
804     Ai1j=dwatn(i+1,j)*coswta
805     Bi1j=((heff(i+2,j+1,1)+heff(i+1,j+1,1)+heff(i+1,j,1)
806     < +heff(i+2,j,1))/4.*rhoice*fcor)+(dwatn(i+1,j)*sinwta)
807     Aij1=dwatn(i,j+1)*coswta
808     Bij1=((heff(i+1,j+2,1)+heff(i,j+2,1)+heff(i,j+1,1)
809     < +heff(i+1,j+1,1))/4.*rhoice*fcor)+(dwatn(i,j+1)*sinwta)
810 c
811     XX(i,j)=((Aij*uice(i,j,1)-Bij*vice(i,j,1))+(Aij1
812     < *uice(i,j+1,1)-Bij1*vice(i,j+1,1)))/2.
813     YY(i,j)=((Aij*vice(i,j,1)+Bij*uice(i,j,1))+(Ai1j
814     < *vice(i+1,j,1)+Bi1j*uice(i+1,j,1)))/2.
815   enddo
816   enddo
817 c
818 c apply B-Grid velocity mask
819 c
820   do j=1,ny
821   do i=1,nx
822     uice(i,j,1)=uice(i,j,1)*uvmb(i,j)
823     vice(i,j,1)=vice(i,j,1)*uvmb(i,j)
824   enddo
825   enddo
826 c
827   return
828   end
829 c

```

```

830 c -----
831      subroutine FREED
832 c -----
833 c calculates free drift velocity field in C-grid by under-relaxation.
834 c There may be more efficient scheme, but this seems to work reliably.
835 c
836 c
837      implicit real*8(a-h,o-z)
838      parameter (nx=30,ny=60)
839      common/one/XX(nx,ny),YY(nx,ny),Ax(nx,ny),Ay(nx,ny),
840      <Bx(nx,ny),By(nx,ny),uice(nx,ny,3),vice(nx,ny,3),
841      <heff(nx+1,ny+1,3),area(nx+1,ny+1,3),presur(nx+1,ny+1),
842      <xmpress(nx+1,ny+1),heffm(nx+1,ny+1),uvmx(nx,ny),
843      <uvmy(nx,ny),uvmb(nx,ny)
844      common/two/deltax,deltay,deltat
845 c
846      nmax=500
847      tol=0.00001
848      w=0.5
849 c
850 c relaxation scheme to find velocity field given pressure field
851 c
852      do kk=1,nmax
853 c
854      do j=1,ny
855      do i=1,nx
856          uice(i,j,2)=uice(i,j,1)
857          vice(i,j,2)=vice(i,j,1)
858      enddo
859      enddo
860 c
861      do j=2,ny-1
862      do i=2,nx-1
863          dpdx=(presur(i+1,j+1)-presur(i,j+1))/deltax
864          dpdy=(presur(i+1,j+1)-presur(i+1,j))/deltay
865          Bvbar=0.25*Bx(i,j)*(vice(i-1,j,1)+vice(i,j,1)
866          <      +vice(i-1,j+1,1)+vice(i,j+1,1))
867          Bubar=0.25*By(i,j)*(uice(i+1,j,1)+uice(i,j,1)
868          <      +uice(i,j-1,1)+uice(i+1,j-1,1))
869          R1=dpdx-XX(i,j)-Bvbar
870          R2=dpdy-YY(i,j)+Bubar
871          uice(i,j,1)=(w*(-1./Ax(i,j))*R1
872          <      +(1.-w)*uice(i,j,2))*uvmx(i,j)
873          vice(i,j,1)=(w*(-1./Ay(i,j))*R2
874          <      +(1.-w)*vice(i,j,2))*uvmy(i,j)
875      enddo
876      enddo
877 c
878      errm=0.
879      do j=1,ny
880      do i=1,nx
881          erru=abs(uice(i,j,1)-uice(i,j,2))

```

```

882     errv=abs(vice(i,j,1)-vice(i,j,2))
883     if(erru.gt.errm)errm=erru
884     if(errv.gt.errm)errm=errv
885   enddo
886   enddo
887   if(errm.le.tol) go to 999
888 c
889   enddo
890   if(kk.ge.nmax) print*, 'max number of iterations exceeded
891   <in free'
892 c
893   999 print*, '# of iterations in free =',kk
894 c
895 c apply C-Grid velocity mask
896 c
897   do j=1,ny
898     do i=1,nx
899       uice(i,j,1)=uice(i,j,1)*uvmx(i,j)
900       vice(i,j,1)=vice(i,j,1)*uvmy(i,j)
901     enddo
902   enddo
903 c
904   return
905 end
906 c
907 c -----
908 subroutine GROWTH(kadd)
909 c -----
910 c calculates thermodynamic growth rate, in fixed grid for each
911 c thickness category, then calculates new thickness distribution
912 c for each particle using an upstream-differencing scheme.
913 c
914 c
915 implicit real*8(a-h,o-z)
916 parameter (nx=30,ny=60)
917 parameter (kmax=17500, nl=6, ndes=nl+4)
918 common/one/XX(nx,ny),YY(nx,ny),Ax(nx,ny),Ay(nx,ny),
919 <Bx(nx,ny),By(nx,ny),uice(nx,ny,3),vice(nx,ny,3),
920 <heff(nx+1,ny+1,3),area(nx+1,ny+1,3),presur(nx+1,ny+1),
921 <xmpress(nx+1,ny+1),heffm(nx+1,ny+1),uvmx(nx,ny),
922 <uvmy(nx,ny),uvmb(nx,ny)
923 common/two/deltax,deltay,deltat
924 common/three/dwatn(nx,ny),dairn(nx,ny),gwatx(nx,ny),
925 <gwaty(nx,ny),gairx(nx,ny),gairy(nx,ny),wta,ata,fcor,rhoice
926 common/four/tair(nx+1,ny+1),qa(nx+1,ny+1),
927 <flo(nx+1,ny+1),fsh(nx+1,ny+1),ug(nx+1,ny+1)
928 common/six/tmix(nx+1,ny+1),tice(nx+1,ny+1),hneg(nx+1,ny+1),
929 <fw(nx+1,ny+1)
930 common/pic/part(ndes,kmax),part0(2,kmax),aunit,hunit,
931 <hb(nl+1),hc(nl),knum
932 dimension hice(nx+1,ny+1),fo(nx+1,ny+1),fhtemp(nx+1,ny+1),
933 <f(nx+1,ny+1,nl),Flux(nl+1)

```

```

934      dimension asum(nx+1,ny+1),ncount(nx+1,ny+1),dA(nx+1,ny+1)
935  c
936      zero=0.0D+00
937      epsil=1.D-12          !small number used to avoid dividing by zero
938  c
939  c specify inverse mixed-layer depth
940  c
941      hmixi=1.0D+00/60.0D+00
942  c
943  c calculate mixed-layer temperature
944  c
945      do j=1,ny+1
946          do i=1,nx+1
947              tmix(i,j)=72.0764D+00*hmixi*hneg(i,j)+271.2D+00
948          enddo
949      enddo
950  c
951  c calculate wind speed for use in sensible and latent heat flux
952  c
953      DO 202 J=2,NY
954      DO 202 I=2,NX
955      if(I.eq.nx+1) li=1
956      U1=0.25*(GAIRX(I-1,J-1)+GAIRX(I-1,J)
957      1+GAIRX(I,J-1)+GAIRX(I,J))
958      V1=0.25*(GAIKY(I-1,J-1)+GAIKY(I-1,J)
959      1+GAIKY(I,J-1)+GAIKY(I,J))
960      UG(I,J)=SQRT(U1**2+V1**2)*HEFFM(I,J)+3.0*(1.0-HEFFM(I,J))
961  202 CONTINUE
962  c
963  c...do energy budget calculations to get growth rates for each
964  c...thickness category
965  c
966  c.....first get open water growth rate, fo
967      do j=1,ny+1
968          do i=1,nx+1
969              hice(i,j)=zero
970          enddo
971      enddo
972      call BUDGET(hice,fo,-1)
973  c.....now get growth rates for other thickness categories, f(l)
974      do l=1,nl
975          do j=1,ny+1
976              do i=1,nx+1
977                  hice(i,j)=hc(l)
978              enddo
979          enddo
980          call BUDGET(hice,fhtemp,1)
981          do j=1,ny+1
982              do i=1,nx+1
983                  f(i,j,l)=fhtemp(i,j)
984              enddo
985          enddo

```

```

986      enddo
987  c
988  c...check if grid cell contains no particles and add KADD new
989  c  particles if freezing (new particles have zero volume and
990  c  area). New particles arranged in a circle of radius 0.3 grid
991  c  spacings around the centre of the grid cell. Note that no new
992  c  particles are added under melting conditions - heat simply
993  c  goes into mixed layer.
994  c
995      do j=1,ny+1
996          do i=1,nx+1
997              ncount(i,j)=0
998          enddo
999      enddo
1000 c....count number of particles in each grid cell
1001     do k=1,knum
1002         i=nint(part(1,k))
1003         j=nint(part(2,k))
1004         ncount(i,j)=ncount(i,j)+1
1005     enddo
1006 c....if no particles exist, add new ones
1007     do j=1,ny+1
1008         do i=1,nx+1
1009             if(ncount(i,j).lt.1.and.fo(i,j).gt.zero) then
1010                 if(heffm(i,j).gt.0.5D+00) then
1011                     do kcmt=1,kadd
1012                         knum=knum+1
1013                         ncount(i,j)=ncount(i,j)+1
1014                         if(knum.gt.kmax) go to 999
1015                         xnp=0.3*cos(float(kcmt-1)*2.*3.14159/float(kadd))
1016                         ynp=0.3*sin(float(kcmt-1)*2.*3.14159/float(kadd))
1017                         part(1,knum)=float(i)+xnp
1018                         part(2,knum)=float(j)+ynp
1019                         part(3,knum)=zero
1020                         part(4,knum)=zero
1021                     do l=1,nl
1022                         part(l+4,knum)=zero
1023                     enddo
1024                 enddo
1025             endif
1026         endif
1027     enddo
1028 enddo
1029 c
1030 c...calculate ice-covered area in each grid cell
1031 c
1032     call SUMIJ(asum,4,epsil)
1033 c
1034 c...calculate change in g(h) for each particle as a result of growth or
1035 c...melt by performing upstream-differencing advection in thickness space
1036 c
1037     do k=1,knum

```

```

1038      i=nint(part(1,k))
1039      j=nint(part(2,k))
1040 c.....first calculate exchanges amongst thickness categories
1041      Flux(1)=zero
1042      Flux(nl+1)=zero
1043      do l=2,nl
1044          Flux(l)=max(zero,f(i,j,l-1))*part(l+4-1,k) +
1045          < min(zero,f(i,j,l))*part(l+4,k)
1046          Flux(l)=Flux(l)/(hc(l)-hc(l-1))
1047      enddo
1048      do l=1,nl
1049          dfgdh=Flux(l+1)-Flux(l)
1050          part(l+4,k)=part(l+4,k)-dfgdh*deltat
1051      enddo
1052 c.....special case: growth of thickest category (taken to increase
1053 c.....ice-covered area)
1054      deltag=deltat*max(zero,f(i,j,nl))*part(nl+4,k)/hc(nl)
1055      part(nl+4,k)=part(nl+4,k)+deltag
1056      dAgt=deltag*part(4,k)*aunit
1057 c.....special case: melt of thinnest category (which reduces
1058 c.....ice-covered area)
1059      deltag=deltat*min(zero,f(i,j,1))*part(1+4,k)/hc(1)
1060      part(1+4,k)=part(1+4,k)+deltag
1061      dAmt=deltag*part(4,k)*aunit
1062 c.....partition open water growth equally amongst particles
1063      Aopen=(deltax*deltay)-asum(i,j)*aunit
1064      dAow=Aopen*max(zero,fo(i,j))*deltat/(float(ncount(i,j))*hc(1))
1065      part(1+4,k)=part(1+4,k)+dAow/(part(4,k)*aunit+epsil)
1066 c.....add up changes in particle area
1067      part(4,k)=part(4,k)+(dAow+dAgt+dAmt)/aunit
1068 c.....now renormalize g(h) for each particle
1069      gsum=zero
1070      do l=1,nl
1071          gsum=gsum+part(l+4,k)
1072      enddo
1073      do l=1,nl
1074          part(l+4,k)=part(l+4,k)*1.D+00/(gsum+epsil)
1075      enddo
1076      enddo
1077 c
1078 c...calculate heat absorbed by mixed-layer (open water 'melt')
1079 c
1080      do j=1,ny+1
1081          do i=1,nx+1
1082              dhneg=-min(zero,fo(i,j))*(deltax*deltay-asum(i,j)*aunit)
1083              < *deltat/(deltax*deltay)
1084              hnegr(i,j)=(hnegr(i,j)+dhneg)*heffm(i,j)
1085          enddo
1086      enddo
1087 c
1088 c...calculate change in particle area due to melt by mixed-layer
1089 c...(assumption is that all such melt is lateral melt and so does

```

```

1090 c...not alter thickness distribution, only particle area)
1091 c
1092 c...initialize array HICE
1093   do j=1,ny+1
1094     do i=1,nx+1
1095       hice(i,j)=epsil !small number to avoid division by zero
1096     enddo
1097   enddo
1098 c...calculate available ice volume per unit area
1099   do k=1,knum
1100     i=nint(part(1,k))
1101     j=nint(part(2,k))
1102     do l=1,nl
1103       hice(i,j)=hice(i,j)+part(l+4,k)*hc(l)*part(4,k)
1104     <           *aunit/(deltax*deltay)
1105   enddo
1106 enddo
1107 c...calculate updated ice-covered area in each grid cell
1108 call sumij(asum,4,epsil)
1109 c...calculate change in area due to mixed-layer melting
1110   do j=1,ny+1
1111     do i=1,nx+1
1112       hdif=hice(i,j)-hneg(i,j)
1113       hneg(i,j)=-min(zero,hdif)
1114       Anew=max(zero,hdif*asum(i,j)/hice(i,j))
1115       dA(i,j)=Anew-asum(i,j)
1116     enddo
1117   enddo
1118 c...partition change in particle area to individual particles
1119   do k=1,knum
1120     i=nint(part(1,k))
1121     j=nint(part(2,k))
1122     part(4,k)=part(4,k)*(1.D+00+dA(i,j)/asum(i,j))
1123   enddo
1124
1125 c*** Diagnostic *** check for negative particle area
1126
1127   do k=1,knum
1128     if(part(4,k).lt.-1.0D-10) then
1129       print*,### particle area < -1.0D-10 ',part(4,k),k
1130     endif
1131   enddo
1132 c*** End Diagnostic
1133 c
1134   return
1135 c
1136 c...print error message if too many particles are created, then stop
1137 c
1138 999 continue
1139   print*

```

```

1140      print*, '!!!!!!'
1141      print*, 'I ATTEMPTED TO CREATE MORE PARTICLES THAN
1142      <ALLOWED BY KMAX!'
1143      print*, '! LOCATION (i,j) ',i,j
1144      print*, '!!!!!!'
1145      print*, ''
1146      stop
1147 c
1148      end
1149 c
1150 c _____
1151      subroutine INTERP4
1152 c _____
1153 c calculates particle volume from thickness distribution and then
1154 c interpolates particle volume and area to fixed Eulerian grid
1155 c using a simple 4-point scheme
1156 c
1157 c
1158      implicit real*8(a-h,o-z)
1159      parameter (nx=30,ny=60)
1160      parameter (kmax=17500, nl=6, ndes=nl+4)
1161      common/one/XX(nx,ny),YY(nx,ny),Ax(nx,ny),Ay(nx,ny),
1162      <Bx(nx,ny),By(nx,ny),uice(nx,ny,3),vice(nx,ny,3),
1163      <heff(nx+1,ny+1,3),area(nx+1,ny+1,3),presur(nx+1,ny+1),
1164      <xpress(nx+1,ny+1),heffm(nx+1,ny+1),uvmx(nx,ny),
1165      <uvmy(nx,ny),uvmb(nx,ny)
1166      common/two/deltax,deltay,deltat
1167      common/pic/part(ndes,kmax),part0(2,kmax),aunit,hunit,
1168      <hb(nl+1),hc(nl),knum
1169      dimension w(2,2)
1170 c
1171 c...initialize some parameters and arrays
1172 c
1173      do j=1,ny+1
1174          do i=1,nx+1
1175              area(i,j,1)=0.
1176              heff(i,j,1)=0.
1177          enddo
1178      enddo
1179 c
1180 c...calculate particle volume from area and thickness distribution
1181 c
1182      do k=1,knum
1183          part(3,k)=0.
1184          do l=1, nl
1185              part(3,k)=part(3,k)+part(l+4,k)*hc(l)
1186          enddo
1187          part(3,k)=part(3,k)*part(4,k)*aunit/hunit
1188      enddo
1189 c
1190 c...interpolate area and volume of particles to fixed grid
1191 c

```

```

1192    do k=1,knum
1193        i=int(part(1,k))
1194        j=int(part(2,k))
1195        dx=part(1,k)-float(i)
1196        dy=part(2,k)-float(j)
1197        w(1,1)=(1.-dx)*(1.-dy)
1198        w(2,1)=(dx)*(1.-dy)
1199        w(1,2)=(1.-dx)*(dy)
1200        w(2,2)=(dx)*(dy)
1201    do n=1,2
1202        do m=1,2
1203            heff(i+m-1,j+n-1,1)=heff(i+m-1,j+n-1,1)+w(m,n)
1204            <           *part(3,k)*hunit/(deltax*deltay)
1205            <           area(i+m-1,j+n-1,1)=area(i+m-1,j+n-1,1)+w(m,n)
1206            <           *part(4,k)*aunit/(deltax*deltay)
1207        enddo
1208    enddo
1209    enddo
1210 c
1211    return
1212 end
1213 c
1214 c -----
1215 subroutine INTERPC(areac,kc)
1216 c -----
1217 c to calculate concentration of a given thickness category interpolated
1218 c onto the fixed Eulerian grid using a simple 4-point scheme
1219 c
1220 c kc indicates the thickness level (level = kc-4 ; i.e. kc=5 indicates
1221 c thickness level 1.)
1222 c
1223 c Note: part(4,k) is the area of the k'th particle while part(kc,k) is
1224 c       the fraction of that area covered by ice in thickness level kc-4.
1225 c       So, part(kc,k)*part(4,k) is the area contributed by the k'th
1226 c       particle to the local concentration of thickness level kc-4.
1227 c
1228     implicit real*8(a-h,o-z)
1229     parameter (nx=30, ny=60)
1230     parameter (kmax=17500, nl=6, ndes=nl+4)
1231     common/two/deltax,deltay,deltat
1232     common/pic/part(ndes,kmax),part0(2,kmax),aunit,hunit,
1233     <hb(nl+1),hc(nl),knum
1234     dimension w(2,2),areac(nx+1,ny+1)
1235 c
1236 c...initialize array
1237 c
1238     do i=1,nx+1
1239         do j=1,ny+1
1240             areac(i,j)=0.
1241         enddo
1242     enddo
1243 c

```

```

1244 c...interpolate concentration to fixed grid
1245     do k=1,knum
1246         i=int(part(1,k))
1247         j=int(part(2,k))
1248         dx=part(1,k)-float(i)
1249         dy=part(2,k)-float(j)
1250         w(1,1)=(1.-dx)*(1.-dy)
1251         w(2,1)=(dx)*(1.-dy)
1252         w(1,2)=(1.-dx)*(dy)
1253         w(2,2)=(dx)*(dy)
1254         do n=1,2
1255             do m=1,2
1256                 areac(i+m-1,j+n-1)=areac(i+m-1,j+n-1)+w(m,n)*part(kc,k)
1257                 *part(4,k)*aunit/(deltax*deltay)
1258             enddo
1259         enddo
1260     enddo
1261     return
1262 end
1263 c
1264 c -----
1265 subroutine NLCAV
1266 c -----
1267 c to calculate and apply non-linear drag
1268 c velocity correction in C-grid
1269 c
1270 c
1271     implicit real*8(a-h,o-z)
1272     parameter (nx=30,ny=60)
1273     common/one/XX(nx,ny),YY(nx,ny),Ax(nx,ny),Ay(nx,ny),
1274     <Bx(nx,ny),By(nx,ny),u(nx,ny,3),v(nx,ny,3),
1275     <heff(nx+1,ny+1,3),area(nx+1,ny+1,3),p(nx+1,ny+1),
1276     <xmpress(nx+1,ny+1),heffm(nx+1,ny+1),uvmx(nx,ny),
1277     <uvmy(nx,ny),uvmb(nx,ny)
1278     common/two/deltax,deltay,deltat
1279 c
1280     iterat=1
1281     maxit=1500
1282     toler=0.0005
1283 c
1284 100 continue
1285 c
1286 c update velocity arrays
1287 c
1288     do j=1,ny
1289         do i=1,nx
1290             u(i,j,2)=u(i,j,1)*uvmx(i,j)
1291             v(i,j,2)=v(i,j,1)*uvmy(i,j)
1292         enddo
1293     enddo
1294 c
1295     do j=1,ny-1

```

```

1296      do i=1,nx-1
1297 c
1298 c do nothing if not an open water grid cell
1299 c
1300   if(heffm(i+1,j+1).lt.0.1) go to 101
1301 c
1302 c do not correct outflow grid cells
1303 c
1304 c calculate divergence (before)
1305 c
1306   D=(v(i,j+1,1)-v(i,j,1))/deltay+(u(i+1,j,1)-u(i,j,1))/deltax
1307 c
1308 c calculate pressure increment
1309 c
1310   delp=0.
1311   delp1=(-D*deltax**2)/(1./Ax(i,j)+1./Ax(i+1,j)
1312 <+1./Ay(i,j)+1./Ay(i,j+1))
1313   if(D.le.0.) delp=delp1
1314 c
1315 c modification for grid cells near boundary
1316 c
1317   t1=min(1.D+00,uvmx(i,j))
1318   t2=min(1.D+00,uvmx(i+1,j))
1319   t3=min(1.D+00,uvmy(i,j))
1320   t4=min(1.D+00,uvmy(i,j+1))
1321   dmlt=4.D+00/(t1+t2+t3+t4)
1322   delp=delp*dmlt
1323 c
1324 c now test pressure against strength
1325 c
1326   pdiff=xmpress(i+1,j+1)-(p(i+1,j+1)+delp)
1327   if(pdiff.lt.0.) delp=xmpress(i+1,j+1)-p(i+1,j+1)
1328 c
1329   if(D.gt.0..and.p(i+1,j+1).le.0.) then
1330     delp=0.
1331   endif
1332   if(D.gt.0..and.p(i+1,j+1).gt.0.) then
1333     delp=(max(delp1,-p(i+1,j+1)))
1334   endif
1335 c
1336 c calculate and apply velocity corrections
1337 c
1338   du1=(-1./(Ax(i,j)*deltax))*delp
1339   du2=(1./(Ax(i+1,j)*deltax))*delp
1340   dv1=(-1./(Ay(i,j)*deltay))*delp
1341   dv2=(1./(Ay(i,j+1)*deltay))*delp
1342   u(i,j,1)=(u(i,j,1)+du1)*uvmx(i,j)
1343   u(i+1,j,1)=(u(i+1,j,1)+du2)*uvmx(i+1,j)
1344   v(i,j,1)=(v(i,j,1)+dv1)*uvmy(i,j)
1345   v(i,j+1,1)=(v(i,j+1,1)+dv2)*uvmy(i,j+1)
1346   p(i+1,j+1)=p(i+1,j+1)+delp
1347 c

```

```

1348 101 continue
1349   enddo
1350   enddo
1351 c
1352 c calculate max error
1353 c
1354   xmaxe=0.
1355   xmaxu=0.
1356   xmaxv=0.
1357   iu=0
1358   iv=0
1359   ju=0
1360   jv=0
1361   do j=1,ny
1362     do i=1,nx
1363       uerr=abs(u(i,j,1)-u(i,j,2))
1364       verr=abs(v(i,j,1)-v(i,j,2))
1365       if(uerr.ge.xmaxe) then
1366         xmaxe=uerr
1367         iu=i
1368         ju=j
1369         xmaxu=uerr
1370       endif
1371       if(verr.ge.xmaxe) then
1372         xmaxe=verr
1373         iv=i
1374         jv=j
1375         xmaxv=verr
1376       endif
1377     enddo
1378   enddo
1379 c
1380 c compare with tolerance
1381 c
1382   if(xmaxe.gt.toler) then
1383     iterat=iterat+1
1384     if(iterat.lt.maxit) then
1385       go to 100
1386     else
1387       print*, ' **** max. iterations exceeded in CAV'
1388       print*, 'iu ju xmaxu ',iu,ju,xmaxu
1389       print*, 'iv jv xmaxv ',iv,jv,xmaxv
1390     endif
1391   endif
1392 c
1393   print*,'# of iteration in cavrelax ',iterat
1394   print*,' max error ',xmaxe
1395 c
1396   return
1397 end
1398 c

```

```

1399 c -----
1400      subroutine PICADVECT
1401 c -----
1402 c Subroutine to advect particles based on given velocity field
1403 c and interpolate particle volume to Eulerian grid.
1404 c Based in part on model described by Pavia and Cushman-Roisin,
1405 c J. Geophys. Res., 93, 3554-3562, 1988.
1406 c
1407 c
1408      implicit real*8(a-h,o-z)
1409      parameter (nx=30,ny=60)
1410      parameter (kmax=17500, nl=6, ndes=nl+4)
1411      common/one/XX(nx,ny),YY(nx,ny),Al(nx,ny),Ap(nx,ny),
1412      <Bl(nx,ny),Bp(nx,ny),u(nx,ny,3),v(nx,ny,3),
1413      <heff(nx+1,ny+1,3),area(nx+1,ny+1,3),p(nx+1,ny+1),
1414      <xmpress(nx+1,ny+1),heffm(nx+1,ny+1),uvmx(nx,ny),
1415      <uvmy(nx,ny),uvmb(nx,ny)
1416      common/two/deltax,deltay,deltat
1417      common/pic/part(ndes,kmax),part0(2,kmax),aunit,hunit,
1418      <hb(nl+1),hc(nl),knum
1419      dimension w(2,2)
1420 c
1421 c part(n,k) -> particle information array
1422 c knum=number of particles
1423 c ndes=number of particle descriptors
1424 c      n=1 -> X location
1425 c      n=2 -> Y location
1426 c part0(n,k) -> particle locations from previous time step
1427 c
1428 c...initialize some arrays and constants
1429 c
1430 c...advection particles to new location using forward Euler time step
1431 c
1432      do k=1,knum
1433 c...first interpolate velocity to particle location
1434      i=int(part(1,k))
1435      j=int(part(2,k))
1436      dx=part(1,k)-float(i)
1437      dy=part(2,k)-float(j)
1438      uatk=0.
1439      vatk=0.
1440      w(1,1)=(1.-dx)*(1.-dy)
1441      w(2,1)=(dx)*(1.-dy)
1442      w(1,2)=(1.-dx)*(dy)
1443      w(2,2)=(dx)*(dy)
1444      do n=1,2
1445          do m=1,2
1446              uatk=uatk+0.5*w(m,n)*(u(i+m-2,j+n-2,1)
1447              <           +u(i+m-1,j+n-2,1))
1448              vatk=vatk+0.5*w(m,n)*(v(i+m-2,j+n-2,1)
1449              <           +v(i+m-1,j+n-1,1))
1450      enddo

```

```

1451      enddo
1452 c...update particle location
1453      part(1,k)=part0(1,k)+deltat*uatk/deltax
1454      part(2,k)=part0(2,k)+deltat*vatk/deltay
1455 c...don't move particle if new location is outside of domain
1456 c (this keeps particles from crossing solid boundaries)
1457      i=nint(part(1,k))
1458      j=nint(part(2,k))
1459      if(heffm(i,j).lt.0.1) then
1460          part(1,k)=part0(1,k)
1461          part(2,k)=part0(2,k)
1462      endif
1463      enddo
1464 c
1465      return
1466      end
1467 c
1468 c _____
1469      subroutine PICDEFORM
1470 c _____
1471 c calculates change in particle area after advection,
1472 c so that interpolated compactness does not exceed unity, then
1473 c performs redistribution of thickness as a result of ridging.
1474 c * Area change is proportional to particle area.
1475 c * Called after GROWTH so that excess area created by open water
1476 c growth is also accounted for.
1477 c
1478 c
1479      implicit real*8(a-h,o-z)
1480      parameter (nx=30,ny=60)
1481      parameter (kmax=17500, nl=6, ndes=nl+4)
1482      common/one/XX(nx,ny),YY(nx,ny),Ax(nx,ny),Ay(nx,ny),
1483      <Bx(nx,ny),By(nx,ny),uice(nx,ny,3),vice(nx,ny,3),
1484      <heff(nx+1,ny+1,3),area(nx+1,ny+1,3),presur(nx+1,ny+1),
1485      <xpress(nx+1,ny+1),heffm(nx+1,ny+1),uvmx(nx,ny),
1486      <uvmy(nx,ny),uvmb(nx,ny)
1487      common/two/deltax,deltay,deltat
1488      common/pic/part(ndes,kmax),part0(2,kmax),aunit,hunit,
1489      <hb(nl+1),hc(nl),knum
1490      dimension adiff(kmax)
1491 c
1492 c...specify some parameters
1493 c
1494      epsil=1.D-12      !small number to avoid dividing by zero
1495 c
1496 c...find compactness values by interpolation
1497 c
1498      call interp4
1499 c
1500 c...find required area reduction for each particle
1501 c
1502      do k=1,knum

```

```

1503      i=int(part(1,k))
1504      j=int(part(2,k))
1505      am=0.D+00
1506      do n=1,2
1507          do m=1,2
1508              am=max(area(i+m-1,j+n-1,1),am)
1509          enddo
1510      enddo
1511      adiff(k)=part(4,k)-part(4,k)/max(1.D+00,am)
1512 c.....modify particle area
1513      part(4,k)=part(4,k)/max(1.D+00,am)
1514      adiff(k)=adiff(k)/(part(4,k)+epsil)
1515      enddo
1516 c
1517 c...calculate additional area reduction due to shear deformation
1518 c
1519 c*** no shear ridging included in present version
1520 c
1521 c
1522 c...do ridge redistribution
1523 c
1524     call RIDGE(adiff)
1525 c
1526 c***diagnostic - check where area exceeds 1
1527 c
1528     call interp4
1529     do j=1,ny+1
1530     do i=1,nx+1
1531     if(area(i,j,1).gt.1.000001) then
1532         print*,'
1533         print*, '**** in picdeform **** '
1534         print*,'
1535         print*,i,j,area',i,j,area(i,j,1)
1536         print*,'
1537     endif
1538     enddo
1539     enddo
1540 c
1541 c***end diagnostic
1542 c
1543 c
1544     return
1545 end
1546 c
1547 c _____
1548     subroutine RIDGE(adiff)
1549 c
1550 c performs ridge redistribution given area change ADIFF. ridging functions
1551 c parameterized as in Thorndike et al. (J.Geophys.Res.,80,4501-4513,1875).
1552 c Discrete redistribution scheme follows Hibler
1553 c (Mon.Wea.Rev.,108,1943-1973,1980 - Appendix C.3).
1554 c

```

```

1555 c
1556      implicit real*8(a-h,o-z)
1557      parameter (nx=30,ny=60)
1558      parameter (kmax=17500, nl=6, ndes=nl+4)
1559      common/one/XX(nx,ny),YY(nx,ny),Ax(nx,ny),Ay(nx,ny),
1560      <Bx(nx,ny),By(nx,ny),uice(nx,ny,3),vice(nx,ny,3),
1561      <heff(nx+1,ny+1,3),area(nx+1,ny+1,3),presur(nx+1,ny+1),
1562      <xpress(nx+1,ny+1),heffm(nx+1,ny+1),uvmx(nx,ny),
1563      <uvmy(nx,ny),uvmb(nx,ny)
1564      common/two/deltax,deltay,deltat
1565      common/pic/part(ndes,kmax),partO(2,kmax),aunit,hunit,
1566      <hb(nl+1),hc(nl),knum
1567      common/redist/rk,Gstar
1568      dimension adiff(kmax),G(nl+1,kmax),lm(kmax),b(nl+1,kmax),
1569      <gamma(nl,nl),a(nl,kmax),dA(kmax),hstar(kmax)
1570 c
1571 c...some preliminaries
1572 c
1573      do k=1,knum
1574 c.....initialize some arrays
1575      lm(k)=0
1576      dA(k)=0.D+00
1577      hstar(k)=0.D+00
1578      do l=1, nl
1579          a(l,k)=0.D+00
1580      enddo
1581      do l=1, nl+1
1582          b(l,k)=0.D+00
1583          G(l,k)=0.D+00
1584      enddo
1585 c.....increase g(h) values to reflect convergence
1586      do l=1, nl
1587          part(l+4,k)=part(l+4,k)*(1.D+00+adiff(k))
1588      enddo
1589  enddo
1590 c
1591 c...construct discrete redistribution process
1592 c
1593      do l=1, nl
1594          do m=1, nl-1
1595              gamma(l,m)=0.D+00
1596              if(hb(m).le.rk*hc(l).and.hb(m+1).gt.rk*hc(l)) then
1597                  gamma(l,m)=1.D+00/rk
1598              endif
1599          enddo
1600          gamma(l,nl)=0.D+00
1601          if(hc(nl-1).lt.rk*hb(l))gamma(l,nl)=1.D+00/rk
1602      enddo
1603 c.....normalize gamma
1604      do l=1, nl
1605          hl=0.D+00
1606          do m=1, nl

```

```

1607      hl=hl+gamma(l,m)*hc(m)
1608      enddo
1609      do m=1,nl
1610          gamma(l,m)=gamma(l,m)*hc(l)/hl
1611      enddo
1612      enddo
1613 c
1614 c...calculate distribution of ice participating in ridging
1615 c
1616 c.....calculate cumulative distribution function
1617      do k=1,knum
1618          do l=2,nl+1
1619              G(l,k)=G(l-1,k)+part(l+4,k)
1620          enddo
1621      enddo
1622 c.....calculate b(h) at discrete thickness points
1623      do k=1,knum
1624          b(1,k)=2.D+00/Gstar
1625          do l=2,nl+1
1626              b(l,k)=(2.D+00/Gstar)*(1.D+00-G(l,k)/Gstar)
1627              if(b(l,k).lt.0.D+00.and.b(l-1,k).ge.0.D+00) lm(k)=l
1628          enddo
1629      enddo
1630 c.....calculate hstar, thickness at which b(h)=0
1631      do k=1,knum
1632          l=lm(k)
1633          hstar(k)=hb(l-1)-b(l-1,k)*((hb(l)-hb(l-1))/(b(l,k)-b(l-1,k)))
1634      enddo
1635 c.....calculate approximate distribution of ice destroyed by ridging
1636      do k=1,knum
1637          do l=1,lm(k)-2
1638              a(l,k)=part(l+4,k)*0.5D+00*(b(l+1,k)+b(l,k))*(hb(l+1)-hb(l))
1639          enddo
1640          l=lm(k)-1
1641          a(l,k)=part(l+4,k)*0.5D+00*b(l,k)*(hstar(k)-hb(l))
1642          do l=lm(k),nl
1643              a(l,k)=0.D+00
1644          enddo
1645      enddo
1646 c
1647 c...calculate change in area produced by applying redistribution process
1648 c to this distribution (used in normalizing overall redistributor)
1649 c
1650      do k=1,knum
1651          do l=1,nl
1652              do m=1,nl
1653                  dA(k)=dA(k)+a(l,k)*gamma(l,m)
1654              enddo
1655              dA(k)=dA(k)-a(l,k)
1656          enddo
1657      enddo
1658 c

```

```

1659 c...calculate normalization factor so that area change due to
1660 c redistribution is exactly that required for area conservation. Then
1661 c carry out redistribution.
1662 c
1663   do k=1,knum
1664     rnorm=-adiff(k)/(dA(k)+1.D-12)
1665     do l=1,nl
1666       xn=0.D+00
1667       do m=1,nl
1668         xn=xn+a(m,k)*gamma(m,l)
1669       enddo
1670       part(l+4,k)=part(l+4,k)+rnorm*(-a(l,k)+xn)
1671     enddo
1672   enddo
1673 c
1674   return
1675 end
1676 c
1677 c _____
1678 subroutine SHEAR(phi)
1679 c _____
1680 c calculates ice velocity in C-Grid given pressure and shear strength,
1681 c assuming Mohr-Coulomb rheology, using under-relaxation scheme.
1682 c
1683 c
1684 implicit real*8(a-h,o-z)
1685 parameter (nx=30,ny=60)
1686 common/one/XX(nx,ny),YY(nx,ny),Ax(nx,ny),Ay(nx,ny),
1687 <Bx(nx,ny),By(nx,ny),uice(nx,ny,3),vice(nx,ny,3),
1688 <heff(nx+1,ny+1,3),area(nx+1,ny+1,3),presur(nx+1,ny+1),
1689 <xpress(nx+1,ny+1),heffm(nx+1,ny+1),uvmx(nx,ny),
1690 <uvmy(nx,ny),uvmb(nx,ny)
1691 common/two/deltax,deltay,deltat
1692 dimension eta(nx+1,ny+1)
1693 c
1694 nmax=5000
1695 tol=0.00002
1696 sinphi=sin(phi*0.0174533)
1697 etamax=1.D+12
1698 errm=0.
1699 c
1700 c under-relaxation parameter, w
1701 c
1702   w=0.5
1703 c
1704 c find shear viscosity field
1705 c
1706   do j=1,ny+1
1707     do i=1,nx+1
1708       eta(i,j)=0.
1709     enddo
1710   enddo

```

```

1711 c
1712   do j=1,ny-1
1713   do i=1,nx-1
1714     im1=max(i-1,1)
1715     jm1=max(j-1,1)
1716     eta(i+1,j+1)=0.
1717     if(heffm(i+1,j+1).gt.0.) then
1718       e11=(uice(i+1,j,1)-uice(i,j,1))/deltax
1719       e22=(vice(i,j+1,1)-vice(i,j,1))/deltay
1720       sul=0.5*((uice(i,j+1,1)-uice(i,j,1))/deltay
1721       <    +(vice(i,j+1,1)-vice(i-1,j+1,1))/deltax)
1722       <    sur=0.5*((uice(i+1,j+1,1)-uice(i+1,j,1))/deltay
1723       <    +(vice(i+1,j+1,1)-vice(i,j+1,1))/deltax)
1724       <    sll=0.5*((uice(i,j,1)-uice(i,j-1,1))/deltay
1725       <    +(vice(i,j,1)-vice(i-1,j,1))/deltax)
1726       <    slr=0.5*((uice(i+1,j,1)-uice(i+1,j-1,1))/deltay
1727       <    +(vice(i+1,j,1)-vice(i,j,1))/deltax)
1728       ut=uvmb(i,j)+uvmb(i+1,j)+uvmb(i,j+1)+uvmb(i+1,j+1)
1729       e12=(sul+sur+sll+slr)/ut
1730   c calculate principal components
1731   emin=min(e11,e22)
1732   cent=(e11+e22)/2.
1733   Rad=sqrt((cent-emin)**2+e12**2)
1734   eps1=cent+Rad
1735   eps2=cent-Rad
1736   c calculate shear viscosity
1737   ediff=eps1-eps2
1738   shearm=presur(i+1,j+1)*sinphi
1739   etemp=etamax
1740   if(ediff.ne.0.) then
1741     etemp=shearm/abs(ediff)
1742   endif
1743   eta(i+1,j+1)=min(etemp,etamax)
1744   endif
1745   enddo
1746   enddo
1747 c
1748 c relaxation scheme to find velocity field given pressure field
1749 c and shear viscosity field
1750 c
1751   do kk=1,nmax
1752 c
1753   do j=1,ny
1754   do i=1,nx
1755     uice(i,j,2)=uice(i,j,1)
1756     vice(i,j,2)=vice(i,j,1)
1757   enddo
1758   enddo
1759 c
1760   do j=2,ny-1
1761   do i=2,nx-1
1762   if(uvmx(i,j).gt.0..or.uvmy(i,j).gt.0.) then

```

```

1763      dpdx=(presur(i+1,j+1)-presur(i,j+1))/deltax
1764      dpdy=(presur(i+1,j+1)-presur(i+1,j))/deltay
1765      Bvbar=0.25*Bx(i,j)*(vice(i-1,j,1)+vice(i,j,1)
1766      <    +vice(i-1,j+1,1)+vice(i,j+1,1))
1767      Bubar=0.25*By(i,j)*(uice(i+1,j,1)+uice(i,j,1)
1768      <    +uice(i,j-1,1)+uice(i+1,j-1,1))
1769      ht1=heffm(i+1,j+1)+heffm(i,j+1)+heffm(i,j+2)+heffm(i+1,j+2)
1770      eta1=(1./ht1)*(eta(i+1,j+1)+eta(i,j+1)+eta(i,j+2)
1771      <    +eta(i+1,j+2))
1772      ht2=heffm(i+1,j+1)+heffm(i,j+1)+heffm(i,j)+heffm(i+1,j)
1773      eta2=(1./ht2)*(eta(i+1,j+1)+eta(i,j+1)+eta(i,j)
1774      <    +eta(i+1,j))
1775      ht3=heffm(i+1,j+1)+heffm(i+2,j+1)+heffm(i+1,j)+heffm(i+2,j)
1776      eta3=(1./ht3)*(eta(i+1,j+1)+eta(i+2,j+1)+eta(i+1,j)
1777      <    +eta(i+2,j))
1778      ht4=heffm(i+1,j+1)+heffm(i,j+1)+heffm(i+1,j)+heffm(i,j)
1779      eta4=(1./ht4)*(eta(i+1,j+1)+eta(i,j+1)+eta(i+1,j)
1780      <    +eta(i,j))
1781      Rhx1=(eta1/deltay)*(uice(i,j+1,1)/deltay+vice(i,j+1,1)/
1782      <    deltax-vice(i-1,j+1,1)/deltax)-(eta2/deltay)*
1783      <    (-uice(i,j-1,1)/deltay+vice(i,j,1)/deltax-vice(i-1,j,1)/
1784      <    deltax)
1785      Rhy1=(eta3/deltax)*(uice(i+1,j,1)/deltay-uice(i+1,j-1,1)/
1786      <    deltax+vice(i+1,j,1)/deltax)-(eta4/deltax)*(uice(i,j,1)/
1787      <    deltax-uice(i,j-1,1)/deltay-vice(i-1,j,1)/deltax)
1788      Rhx2=(eta(i+1,j+1)/deltax)*(uice(i+1,j,1)/deltax-
1789      <    vice(i,j+1,1)/deltay+vice(i,j,1)/deltay)-(eta(i,j+1)/
1790      <    deltax)*(-uice(i-1,j,1)/deltax-vice(i-1,j+1,1)/deltay+
1791      <    vice(i-1,j,1)/deltay)
1792      Rhy2=(eta(i+1,j+1)/deltay)*(vice(i,j+1,1)/deltay-
1793      <    uice(i+1,j,1)/deltax+uice(i,j,1)/deltax)-(eta(i+1,j)/
1794      <    deltax)*(-vice(i,j-1,1)/deltay-uice(i+1,j-1,1)/deltax+
1795      <    uice(i,j-1,1)/deltax)
1796      Coefx=-Ax(i,j)-(eta1/deltay**2)-(eta2/deltay**2)
1797      <    -(eta(i+1,j+1)/deltax**2)-(eta(i,j+1)/deltax**2)
1798      Coefy=-Ay(i,j)-(eta3/deltax**2)-(eta4/deltax**2)
1799      <    -(eta(i+1,j+1)/deltay**2)-(eta(i+1,j)/deltay**2)
1800      R1=dpdx-XX(i,j)-Bvbar-Rhx1-Rhy2
1801      R2=dpdy-YY(i,j)+Bubar-Rhy1-Rhy2
1802      uice(i,j,1)=(w*(1./Coefx)*R1
1803      <    +(1.-w)*uice(i,j,2))*uvmx(i,j)
1804      vice(i,j,1)=(w*(1./Coefy)*R2
1805      <    +(1.-w)*vice(i,j,2))*uvmy(i,j)
1806      endif
1807      enddo
1808      enddo
1809 c
1810      errm=0.
1811      do j=1,ny
1812      do i=1,nx
1813          erru=abs(uice(i,j,1)-uice(i,j,2))
1814          errv=abs(vice(i,j,1)-vice(i,j,2))

```

```

1815      if(erru.gt.errm)errm=erru
1816      if(errv.gt.errm)errm=errv
1817      enddo
1818      enddo
1819      if(errm.le.tol) go to 999
1820 c
1821      enddo
1822      if(kk.ge.nmax) print*, 'max number of iterations exceeded
1823      <in SHEAR'
1824 c
1825 999 continue
1826      print*,'# of iterations in SHEAR =',kk
1827      print*, '      error in SHEAR =',errm
1828      return
1829      end
1830 c
1831 c -----
1832      subroutine SUMIJ(asum,np,epsil)
1833 c -----
1834 c adds up particle attribute 'part(np,k)' at each (i,j) grid cell
1835 c and returns array 'asum'. 'epsil' is an optional small number
1836 c which may be used to avoid dividing by zero later.
1837 c
1838 c
1839      implicit real*8(a-h,o-z)
1840      parameter (nx=30,ny=60)
1841      parameter (kmax=17500, nl=6, ndes=nl+4)
1842      common/pic/part(ndes,kmax),part0(2,kmax),aunit,hunit,
1843      <hb(nl+1),hc(nl),knum
1844      dimension asum(nx+1,ny+1)
1845 c
1846      do j=1,ny+1
1847          do i=1,nx+1
1848              asum(i,j)=0.
1849          enddo
1850      enddo
1851 c
1852      do k=1,knum
1853          i=nint(part(1,k))
1854          j=nint(part(2,k))
1855          asum(i,j)=asum(i,j)+epsil+part(np,k)
1856      enddo
1857 c
1858      return
1859      end
1860 c
1861 c ----- END -----
1862

```

Appendix 2 - Detailed Description of McPIC Code.

The Fortran code for the McPIC model consists of about 1800 lines (see Appendix 1). While the code contains extensive internal documentation, this section gives a more detailed description of the names of parameters and constants, and explains the purpose of each routine used in the program. Cross-references are provided for equation numbers found in the theoretical section of this report with corresponding line numbers in the appropriate segment of the code.

A test case for the model is included in the code: in this, the model domain consists of a rectangular region of 31 by 61 cells with 20 km grid spacing. This is initialized with ice in a rectangular section, covering the upper left area, with a thickness distribution everywhere similar to that found off Newfoundland in Spring. A constant vortex wind field is then applied, centered near one edge of the ice field as described in more detail below.

Line(s):

76-77 Sets the grid size to 31x61 cells, the number of thickness levels to 6, and limits the total number of particles to 17500. These can be changed to suit a particular application.

101-112 Opens files for input and output data: the model can accept mean annual fields for ocean currents and for oceanic heat flux, initial ice distribution based on satellite or ice-chart data and, at each time step, the fields of geostrophic winds and air temperatures. For the test case included here, no ocean currents are used, the wind is specified as a constant vortex field, and the oceanic heat flux field as increasing from upper right to lower left of the domain.

The model output consists of files for computed ice concentration, thickness, velocity, particle locations and areas, and partial concentration for each of the six thickness categories.

116-141 Specify fixed parameters for the model: number of days in the simulation, number of time steps per day, grid size in metres, number of days in simulation, Coriolis parameter, ice density, the air- and water drag coefficients and turning angles, the compression and shear strength parameters, the ridge thickness and ridge participation parameters, and the boundaries for the thickness categories.

157 Calls subroutine **BND** (lines 552-587) to generate velocity masks from thickness boundary mask specified in data block **COMINT** (lines 702-749). 0 indicates a boundary point, while 1 indicates an ocean point. A two grid cell boundary surrounds the entire domain.

185-218 Initialize particle location and thickness distribution (e.g. from ice maps, or satellite images). In the test case, 9 regularly spaced particles per grid cell are placed in a rectangular subsection of the domain, in cells with $i=3,..17$ and $j=19,..59$. The thickness distribution in those cells is set to approximate the average observed value for March off the Newfoundland Coast (Symonds, Proc. East Coast Workshop on Sea Ice, BIO, 1986).

222 Calls subroutine **INTERP4** (lines 1151-1212) which calculates particle volume from thickness distribution, then interpolates particle volume and area to fixed Eulerian grid.

227 Option to read the annual average ocean current field. No ocean currents are specified for the test case, that is, $gwatx(i,j) = gwaty(i,j) = 0$.

230-241 Specifies the annual average oceanic heat flux. This can be read from a file, but for the test case the field is specified so that the flux increases from -150 Wm^{-2} at the upper left to -250 Wm^{-2} at the lower right.

249 Do loop for number of time steps in the simulation. This loop ends at line 536.

253-260 Calculate the ice strength P_{max} as formulated in equation (9).

262-294 Read the wind data. For the test case included here, a fixed vortex wind field is specified, with maximum winds of 30 ms^{-1} at 30 km from its centre located near the ice edge at $(x,y)=(20.1,30.1)$ in grid coordinates.

296-304 A minimum value is put on wind velocity components ($gairx, gairy$) to avoid later problems with inverse trig functions.

306 Do loop for modified-Euler time-stepping scheme. This loop ends at line 379.

310-319 Average the $uice, vice$ velocities for the second iteration.

321-328 Update ice velocity array in the second step.

330-365 compute drag coefficients for wind (*dairn*) and for water (*dwatn*) at each grid point, which are functions of wind speed and of the ice velocity relative to the water current as in equation (8).

343-353 Optional linear drag coefficients, which are not functions of wind or ice velocity.

355-365 Calculate terms in momentum equation *Ax,Bx* and *Ay,By* (see Flato and Hibler, 1992 for details).

369 Call subroutine **FORCE** (lines 752-828) to calculate velocity-independent forcing field in the B-grid, and free-drift velocity field in the B-grid as initial guess for next step (C-grid free drift).

373 Call subroutine **FREED** (lines 831-905) to calculate the free drift velocity in the C-grid by under-relaxation.

377 Now call subroutine **NLCAV** (lines 1265-1397) to calculate and apply non-linear drag cavitating fluid velocity correction in C-grid, and obtain pressure field.

Note that for linear drag calculation, only one loop is needed over this time-stepping scheme (use "do *kme=1,1*" in line 308).

385 Call subroutine **SHEAR** (lines 1678-1829) to calculate the final ice velocity in the C-grid for given pressure field, assuming Mohr-Coulomb rheology, and shear strength parameter *phi* using an under-relaxation scheme.

387-400 Advect particles using a predictor/corrector scheme that call subroutine **PICADVECT** (lines 1400-1466) to advect particles based on given velocity field, and to interpolate particle volume to Eulerian grid (corresponds to equation (6)).

401 Read in thermodynamic forcing fields. For test case, air temperature, *TAIR*, and the long wave, *FLO*, and shortwave, *FSH*, radiative fluxes are assumed to decrease linearly from south to north (after Symonds (1986) - estimates for March, April and May are included).

- 430 Call subroutine **GROWTH** (lines 908-1148) to calculate thermodynamic growth/decay rate in fixed grid for each thickness category, then calculates the new thickness distribution for each particle using an upstream-differencing method. Thermodynamics is done before computing deformation so that area change due to growth/melt is applied before the constraint that ice covered area fraction ≤ 1 .
- 437 Call subroutine **PICDEFORM** (lines 1469-1545) to calculate the change in particle area after advection, so that interpolated compactness does not exceed unity, then perform redistribution of thickness as a result of ridging. The area change is proportional to particle area.
- 441 Call subroutine **INTERP4** (lines 1151-1212) again, to update the arrays of mean ice thickness and total concentration (see equation (2)).
- 443-457 Compute and print total ice volume and total mixed layer heat storage to standard output.
- 459-483 Interpolate velocity field back to B-grid for plotting and calculate average velocity field.
- 485-523 print fields of total concentration, mean thickness, average ice velocity field, and partial concentration of ice in each thickness category. Also print end-of-day position and area of each particle. There is an option to print these out only at the end of the simulation.
- 513 Call subroutine **INTERPC** to compute partial concentrations for each thickness category from particle thickness distributions and areas.

Subroutines:

552-587 subroutine **BND** - calculates the velocity mask in B-grid and C-grid using the thickness mask *heffm* specified in block data **COMINT** (lines 702-749).

590-699 subroutine **BUDGET** calculates the surface energy budget and hence the ice growth or melt rate (modified subroutine from W.D. Hibler III, based on Heat Budget code in Mon. Wea. Rev., 108, 1943-1973, 1980, Appendix B).

702-749 block data **COMINT** specifies the thickness boundary mask. 0 indicates a boundary point and 1 an ocean point.

752-828 subroutine **FORCE** calculates velocity-independent forcing terms in the B-grid. As a by-product, this subroutine calculates the free drift velocity field in the B-grid which is used as an initial guess to speed up C-grid free drift iteration in subroutine **FREED**.

831-905 subroutine **FREED** calculates free drift velocity field in C-grid by under-relaxation. This scheme, if not efficient, appears to work reliably.

908-1148 subroutine **GROWTH** calculates thermodynamic growth rate in fixed grid for each thickness category, then calculates new thickness distribution for each particle using an upstream-differencing scheme. Calls subroutine **BUDGET** (lines 590-699).

- 941 Specifies (inverse of) mixed-layer depth.
- 945-949 Calculate mixed-layer temperature.
- 953-961 Calculate windspeed for use in sensible and latent heat flux.
- 967-986 Call **BUDGET** to get growth rates for open water and for each thickness category.
- 995-1028 Add new particles if freezing occurs and grid cell has no particles.
- 1032 Call **SUMIJ** to compute ice-covered area in each cell.
- 1037-1076 Calculate change in $g(h)$ for each particle due to growth or melt.
- 1080-1086 Calculate heat absorbed by mixed-layer.
- 1093-1123 Calculate change in particle area due to melt by mixed-layer (assumes all melt reduces only area and not thickness distribution).
- 1127-1146 Diagnostic and error check for negative area or too many particles.

Cross-reference of line numbers and equation numbers:

line(s)	eqn.	line(s)	eqn.
967-972	34	1064	45
974-986	35	1065	46
1041-1047	36	1069-1075	47
1048-1051	37, 38	1080-1086	48
1054	39	1099-1106	49
1056	40	1103	50
1059	41	1113	52
1061	42	1114	53
1063	43	1115-1123	54

1151-1212 subroutine **INTERP4** calculates particle volume from the thickness distribution, then interpolates particle volume and area to the fixed Eulerian grid using a simple 4-point scheme. (line 1205 corresponds to equation 2 page 2, lines 1197-1200 form equation (4) and lines 1182-1188 form equation (12)).

1215-1262 subroutine **INTERPC** calculates the partial concentration of a given thickness category interpolated onto the fixed Eulerian grid using a simple 4-point scheme.

1265-1397 subroutine **NLCAV** calculates and applies non-linear drag cavitating fluid velocity correction in C-grid. This produces a velocity field and pressure field assuming zero shear strength. Shear resistance is incorporated in a later call to subroutine **SHEAR**.

1400-1466 subroutine **PICADVECT** advects particles based on given velocity field and interpolates particle volume to Eulerian grid (lines 1444-1451 correspond to equations (5)). The formulation is based in part on the model described by Pavia and Cushman-Roisin in J. Geophys. Res., 93, 3554-3562, 1988.

1469-1545 subroutine **PICDEFORM** calculates the change in particle area after advection, so that interpolated compactness does not exceed unity, then performs redistribution of thickness as a result of ridging. The area change is proportional to particle area. Called after **GROWTH** so that excess area created by open water growth is also accounted for (lines 1511-1514 correspond to equation (16)).

Calls subroutines **INTERP4** and **RIDGE**.

1548-1575 subroutine **RIDGE** performs ridge redistribution given area change, *adiff*. The ridging functions are parameterized as in Thorndike *et al.* (J.Geophys. Res., 80, 4501-4513, 1975), and the discrete redistribution scheme follows Hibler (Mon. Wea. Rev., 108, 1943-1973, 1980 - Appendix C.3).

Cross-reference of line numbers and equation numbers:

line(s)	eqn.	line(s)	eqn.
1594-1599	18, 22	1650-1657	28
1631-1634	20	1633	29
1624	21, 26	1595-1596	30(1)
1604-1612	23, 31	1610	30(2)
1600-1612	24	1651-1655, 1664-1671	32
1624-1628	27		

1678-1829 subroutine **SHEAR** calculates ice velocities in C-Grid with given pressure and shear strength parameter ϕ , assuming Mohr-Coulomb rheology, and using an under-relaxation scheme.

1832-1859 subroutine **SUMIJ** adds up the values of attribute np of particles $part(np,k)$ for all k within grid cell (i,j) , and returns array $asum$. ϵ_{psil} is an optional small number used to avoid dividing by zero later.

Appendix 3 — McPIC - Test Case: Forcing and Output Plots

As an example of the working and output of the McPIC program, a test case has been included in the code. In this hypothetical application, the model domain is a rectangular region of 31 by 61 cells with a grid spacing of 20 km. The boundary mask, specified in data block "comint" in the code, places a 0 in boundary cells, and a 1 in ocean cells (this can be adapted to include the coastline for any specific region).

At the start of the simulation, an initial rectangle of ice covers the upper two-thirds of the left half of this domain (cells with $i=3..17$ and $j=19..59$). The initial ice thickness distribution in all cells with ice is set to approximate the average observed values for March off the coast of Newfoundland (Symonds, 1986). In specific regional applications, the model can be initialized with data from local ice charts, satellite images or other sources.

The following table shows the boundaries for the thickness categories (in m), the thickness at the centre of each category, and the initial partial concentration for that category:

<u>boundary (m)</u>	<u>centre (m)</u>	<u>partial concentration</u>
0.0	--	0.31
0.15	--	0.25
0.30	--	0.42
1.00	--	0.02
2.50	--	0.0
5.00	--	0.0
10.0	--	

For this test case, no mean ocean currents were used (*i.e.* $g_{\text{watx}}(i,j) = g_{\text{waty}}(i,j) = 0$) and the oceanic heat flux is time invariant and specified to increase from -150 Wm^{-2} at the upper left to -250 Wm^{-2} at the lower right of the domain (loosely based on Symonds, 1986).

The wind field used in this simulation is also kept constant over time: it consists of a vortex centred near the ice edge (located at $x,y=20.1,30.1$ in grid coordinates), with winds increasing linearly from zero at the centre to a maximum of 30 ms^{-1} at 30 km from the centre, then decreasing inversely with distance (see figure 1).

The model simulation was run for a period of 10 days. Most of the initial ice cover has disappeared by day 8 of the simulation as a result of starting with only 0.3 m thick ice. This differs from the result with the single thickness PIC model (Flato 1993b) where ice survived for the entire 10 day period because initial ice thickness there was 0.75 m. While the code has been set up for six different thickness categories, there was no ice in classes 5 and 6 in this simulation.

The output from the simulation consists of a number of files containing the following quantities at the end of each simulation day:

- the mean ice thickness (in m),
- the total ice concentration,
- the mean ice velocity (in ms^{-1}),
- the location of each particle and its area (in km^2), and
- the partial concentration of each of thickness category.

The following pages show plots created from this output:

Figure 1 shows the 31x61 cell model domain and the initial ice cover (rectangle in upper left). The vectors show the specified geostrophic wind field and the contours show the oceanic heat flux (Wm^{-2}). The tickmarks in this and all subsequent plots show the fixed grid spaced at 20 km.

Figure 2 shows the mean ice thickness after 1 day (a) and after 5 days (b) of simulation.

Figure 3 shows the total ice concentration after 1 day (a) and 5 days (b).

Figure 4 shows partial concentrations after 1 day (a) and 5 days (b) for ice in thickness category 1 (0 - 0.15 m).

Figure 5 shows partial concentrations after 1 day (a) and 5 days (b) for ice in thickness category 2 (0.15 - 0.30 m).

Figure 6 shows partial concentrations after 1 day (a) and 5 days (b) for ice in thickness category 3 (0.30 - 1.00 m).

Figure 7 shows partial concentrations after 1 day (a) and 5 days (b) for ice in thickness category 4 (1.00 - 2.50 m).

Figure 8 shows the individual ice particle locations after 1 day (a) and 5 days (b).

The size of the dots has been scaled by the computed area for each particle.

Figure 9 shows the mean ice velocities in each grid cell after 1 day (a) and 5 days (b).

The computed velocity field has been masked to only show the velocities where the total ice concentration exceeds 0.001.

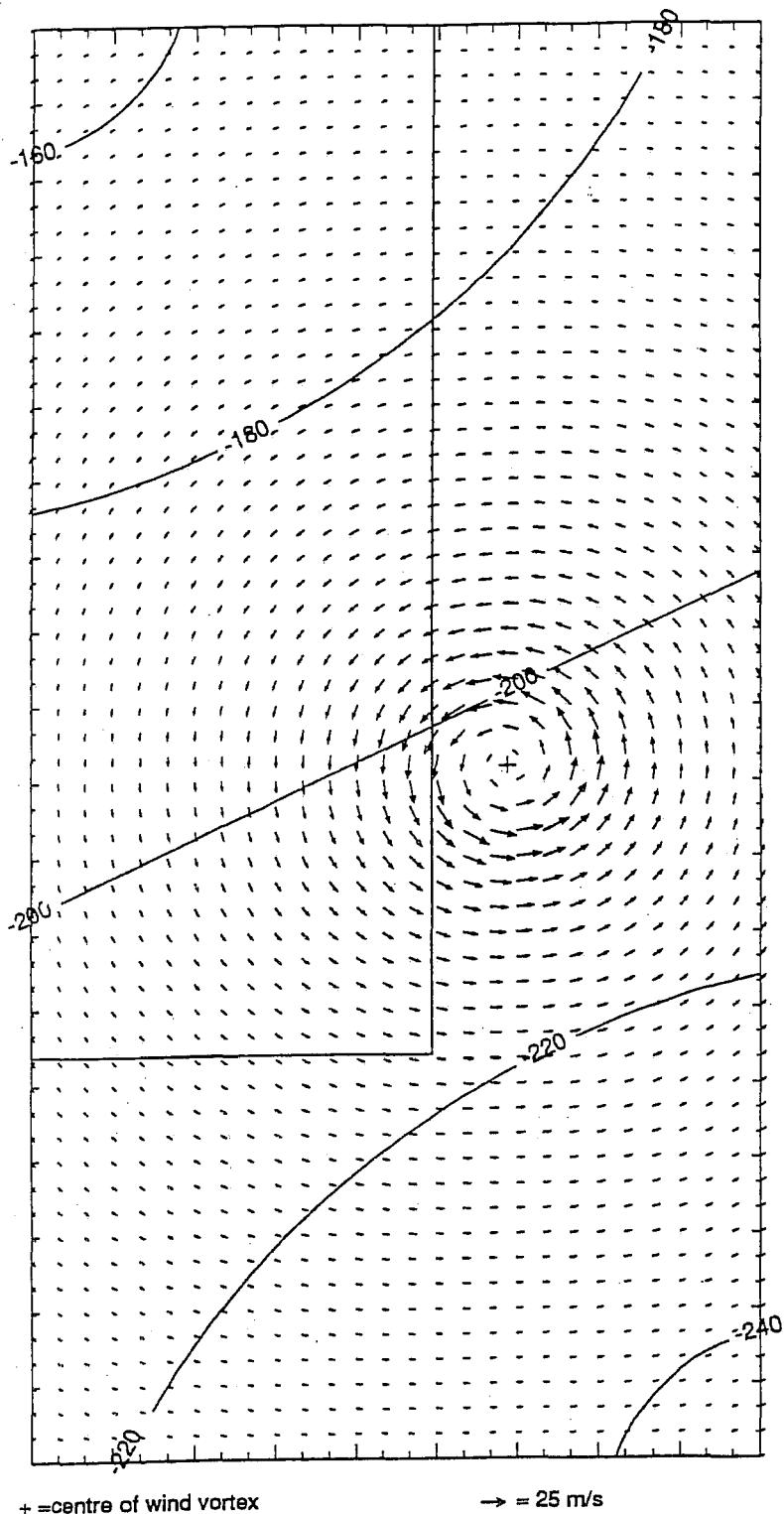


Figure 1. Model domain showing initial ice cover (rectangle in upper left). The vectors show the specified geostrophic wind field and the contours show the oceanic heat flux (Wm^{-2}). The tick marks around the boundary in this and all subsequent plots show the fixed grid spaced at 20 km.

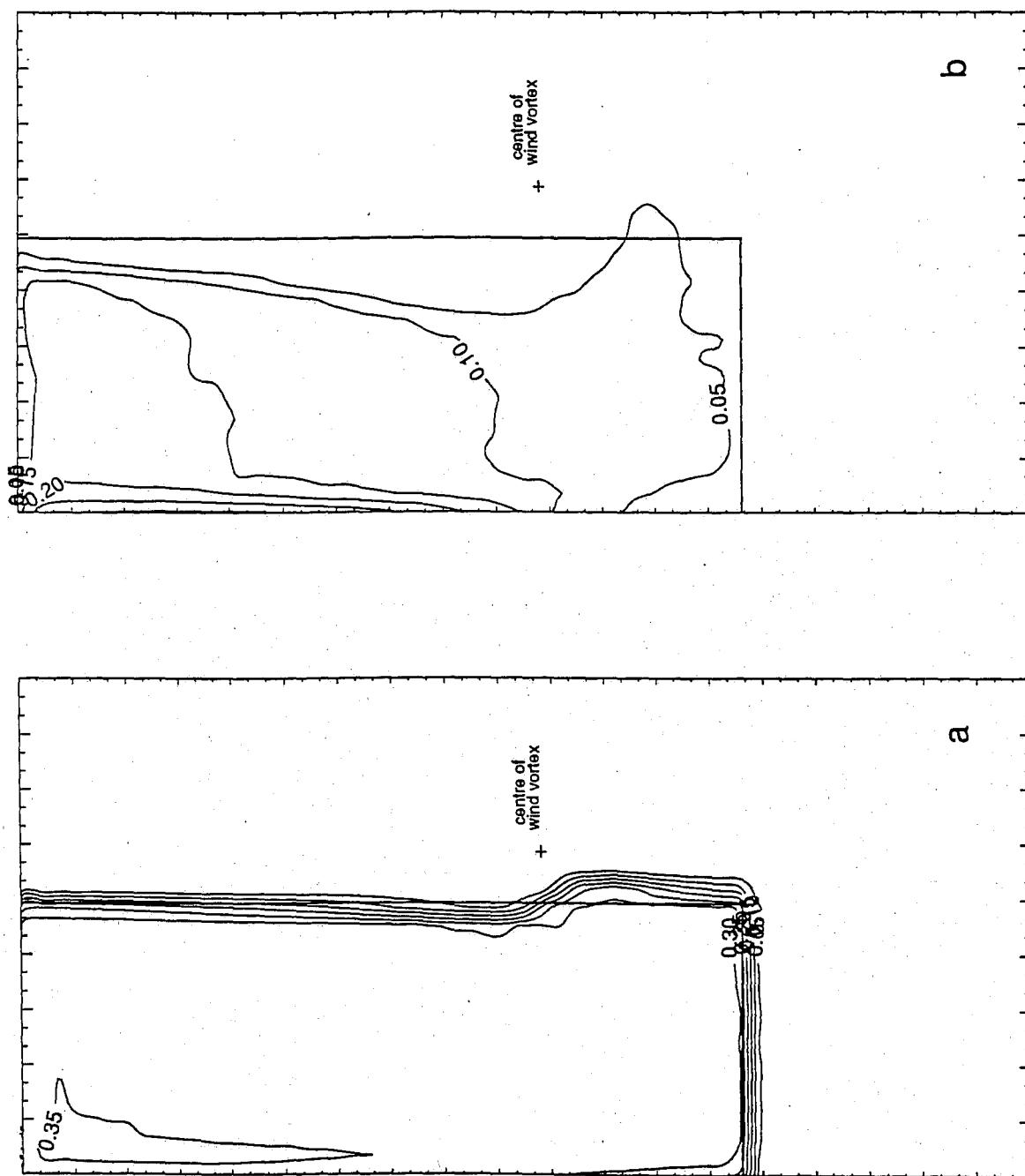


Figure 2. Mean ice thickness: (a) after 1 day and (b) after 5 days.

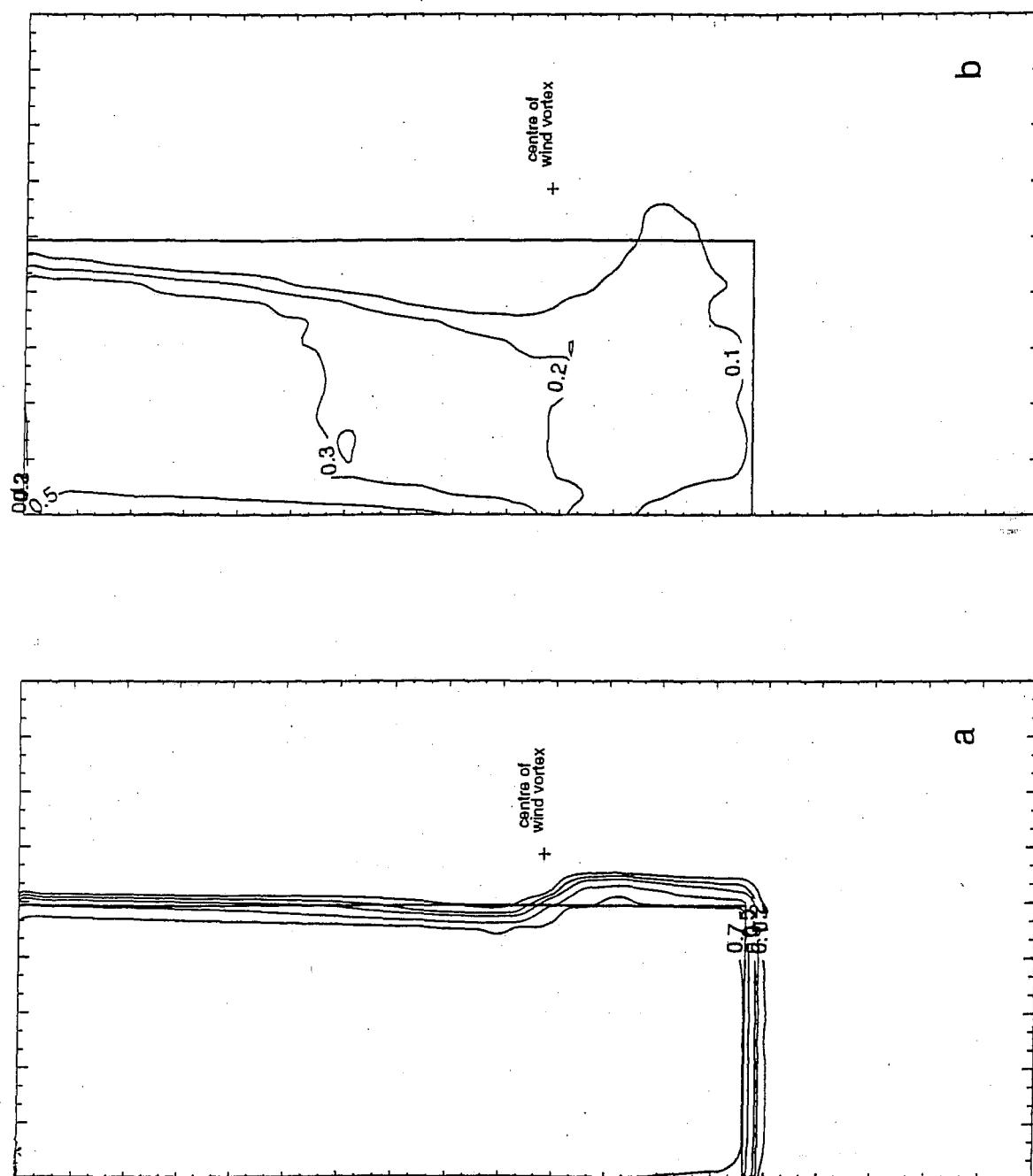


Figure 3. Total ice concentration: (a) after 1 day and (b) after 5 days.

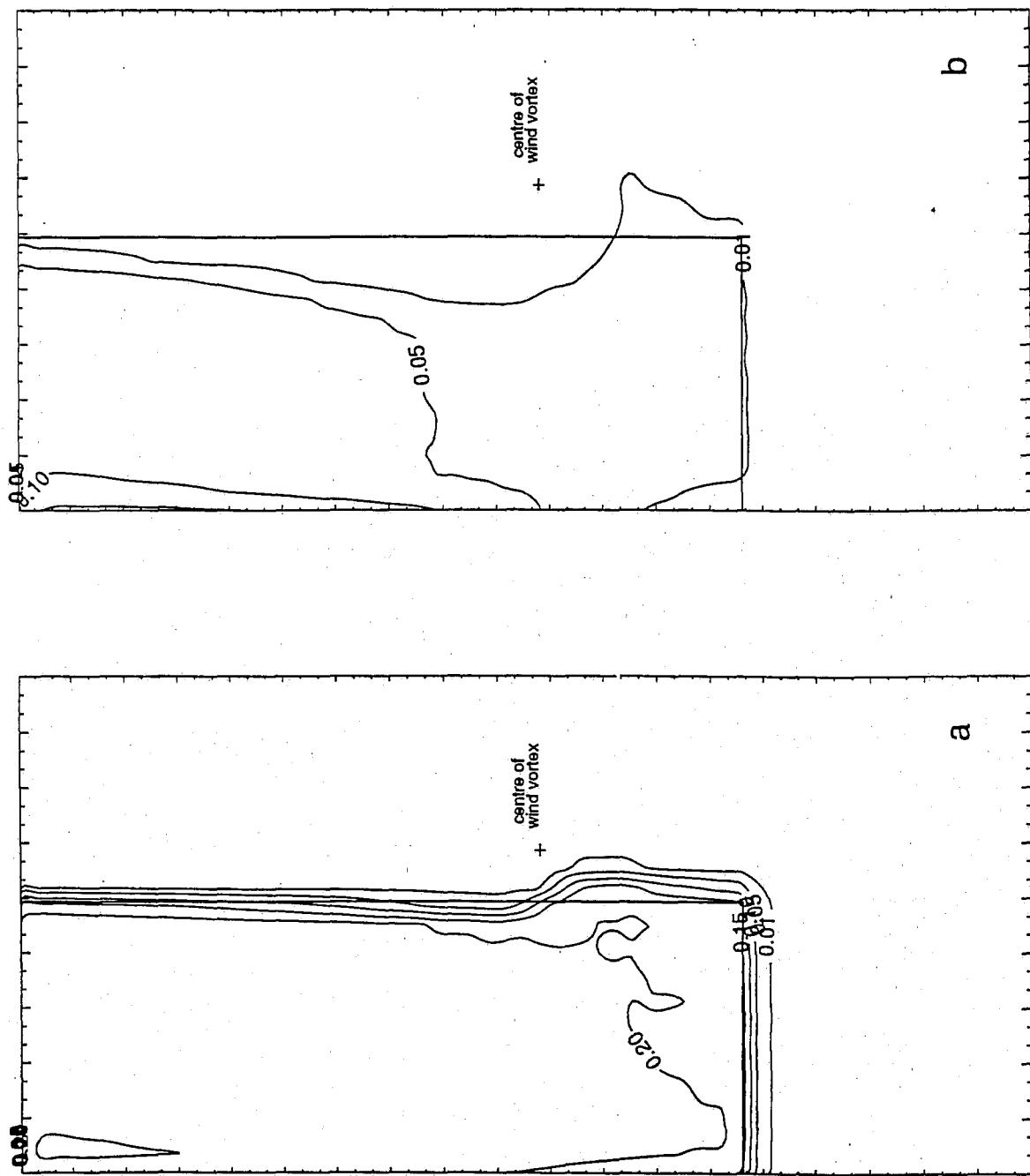


Figure 4. Partial ice concentration of thickness category 1 (0 - 0.15 m):
(a) after 1 day and (b) after 5 days.

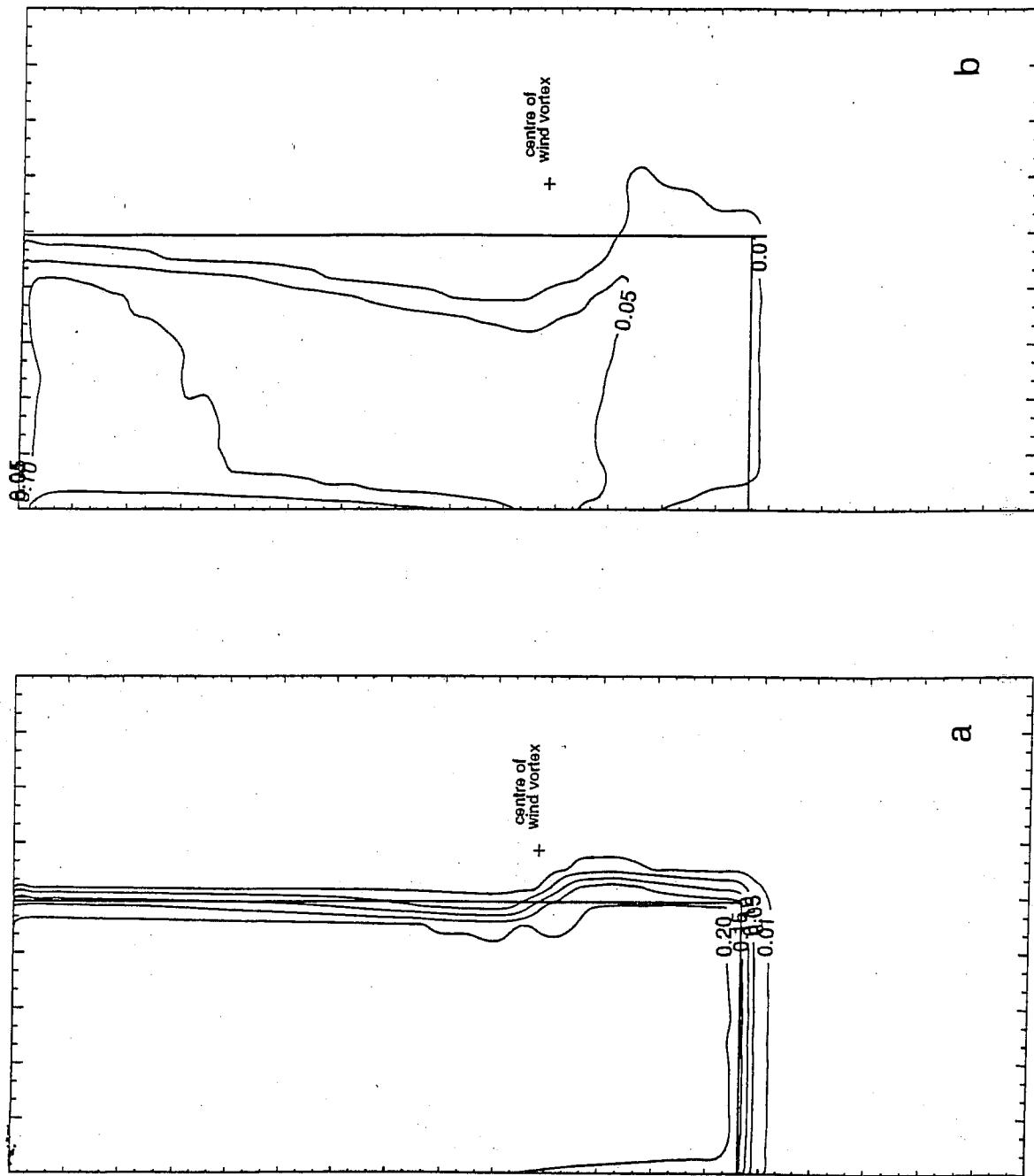


Figure 5. Partial ice concentration of thickness category 2 (0.15 - 0.30 m):
(a) after 1 day and (b) after 5 days.

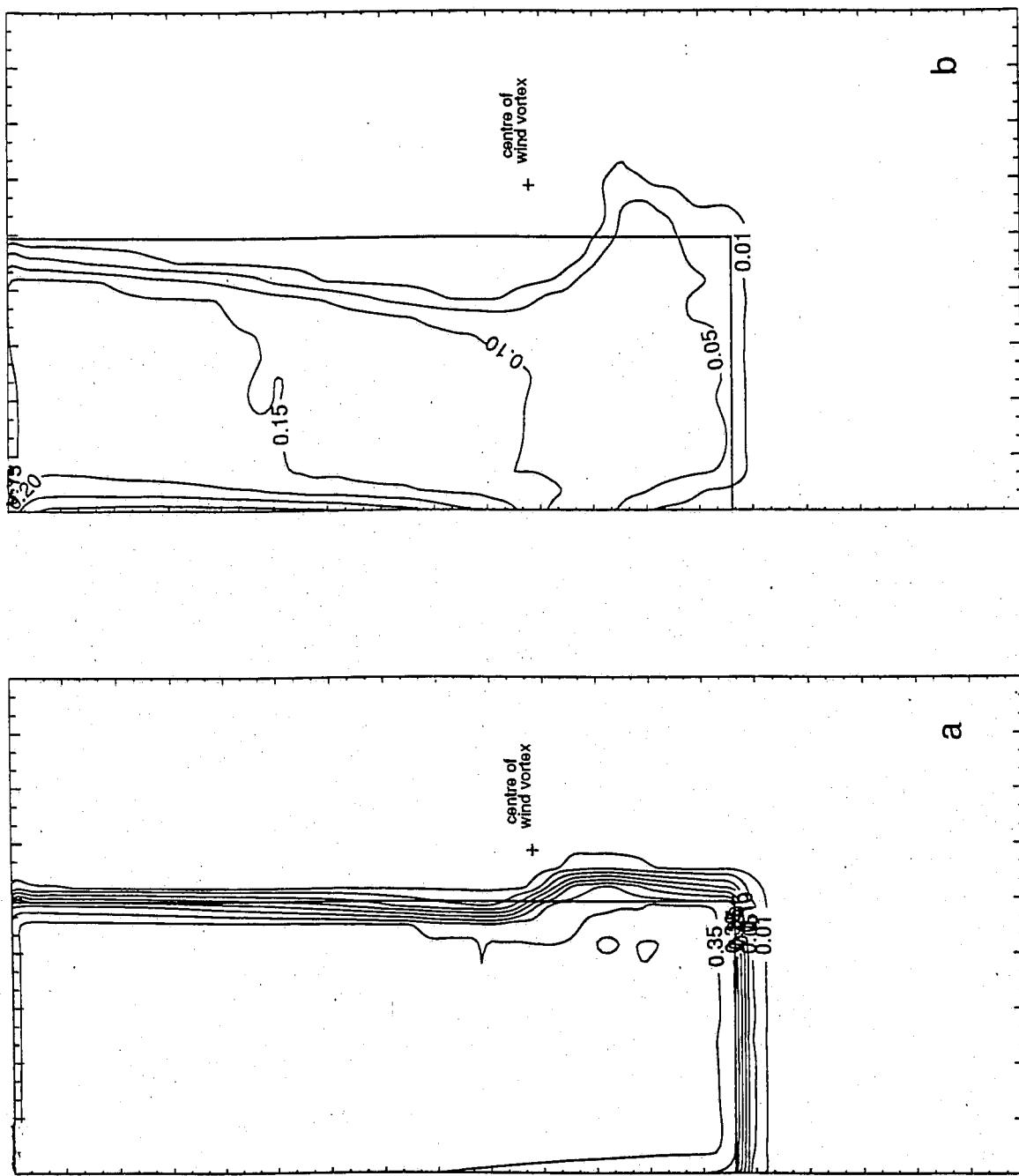


Figure 6. Partial ice concentration of thickness category 3 (0.30 - 1.00 m):
(a) after 1 day and (b) after 5 days.

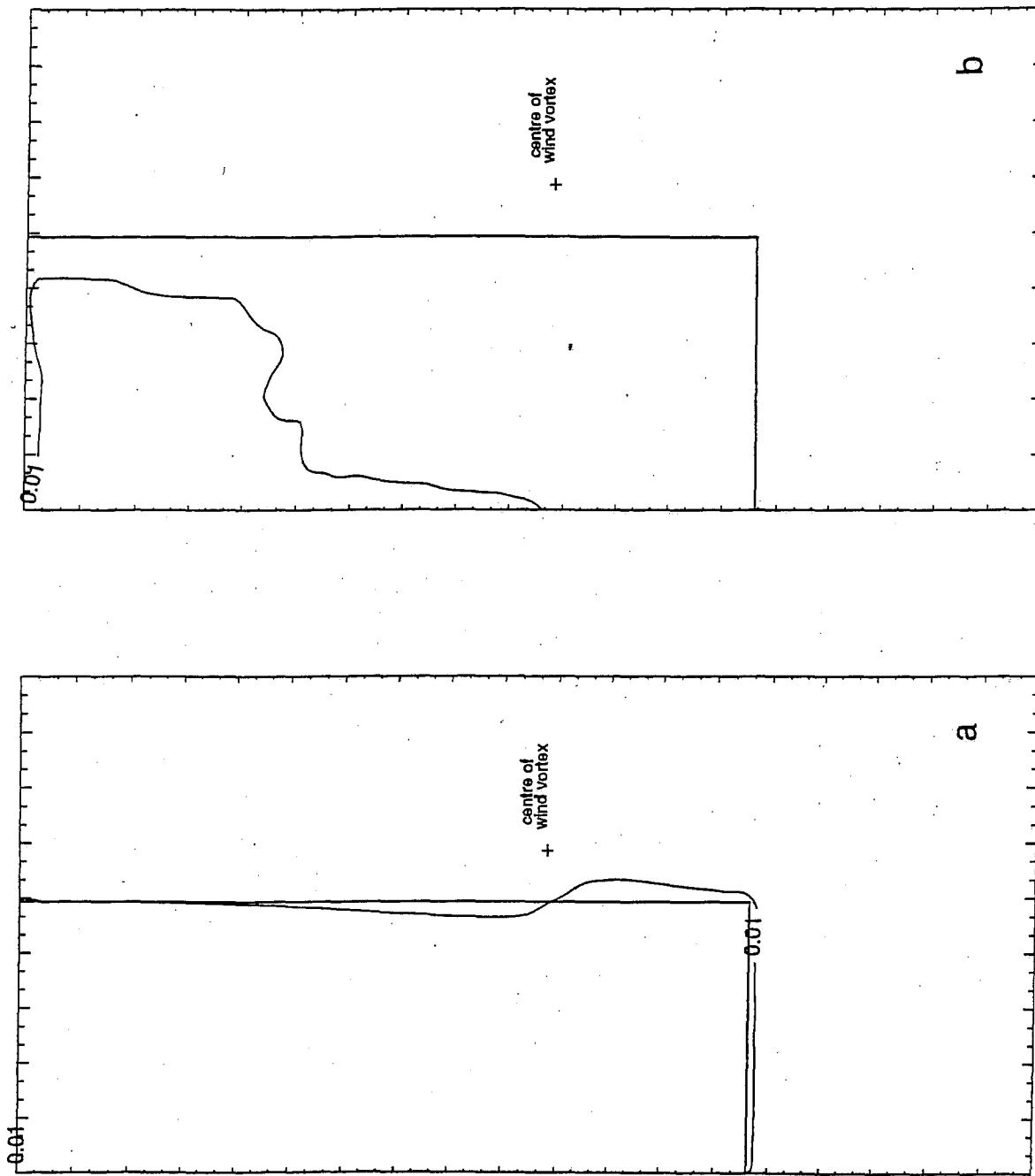


Figure 7. Partial ice concentration of thickness category 4 (1.00 - 2.50 m):
(a) after 1 day and (b) after 5 days.

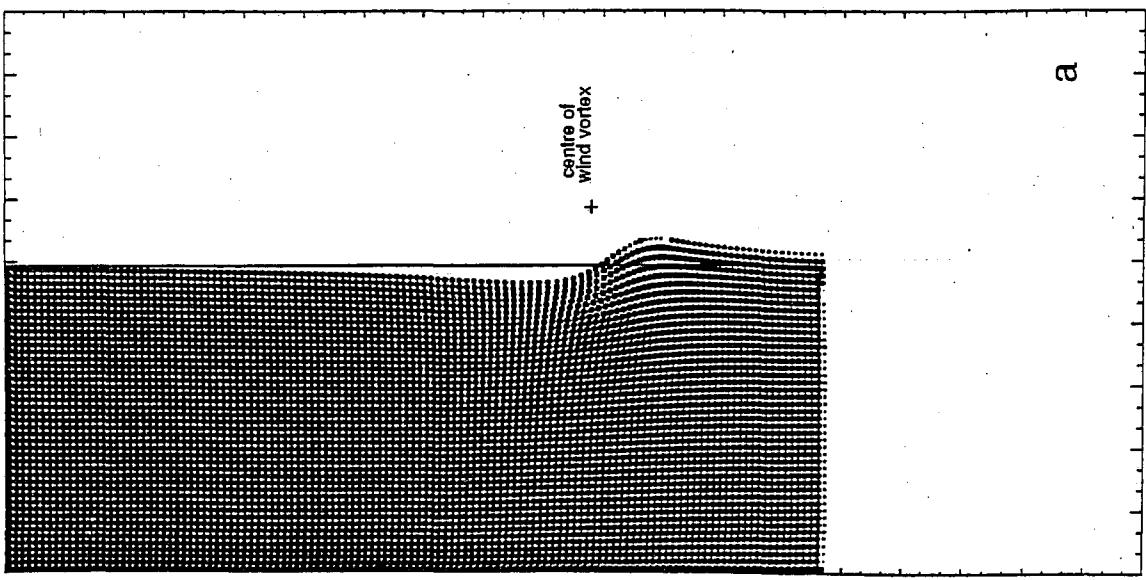
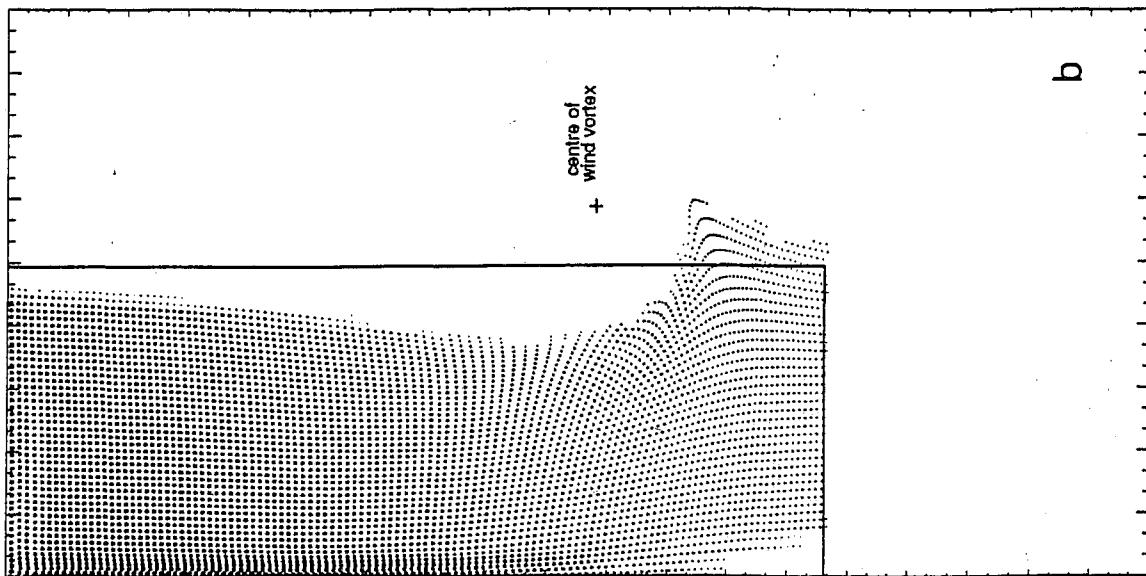


Figure 8. Particle positions: (a) after 1 day and (b) after 5 days.

The dots have been scaled with the particle area.

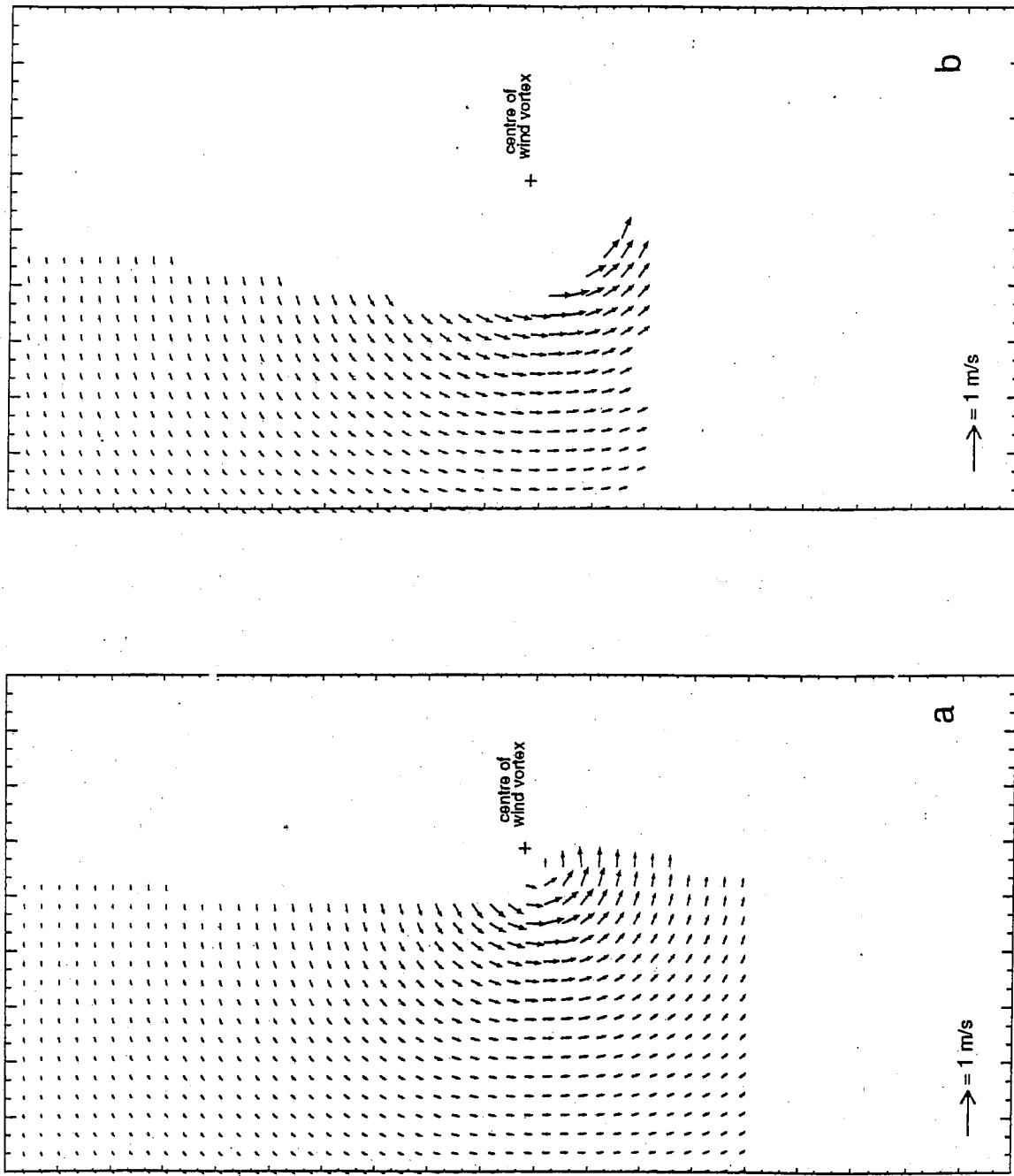


Figure 9. Average ice velocities: (a) after 1 day and (b) after 5 days.
(shown only where total ice concentration exceeds 0.001)