

A parallel Jacobian-free Newton-Krylov solver for a coupled sea ice-ocean model

Martin Losch^{a,*}, Annika Fuchs^a, Jean-François Lemieux^b, Anna Vanselow^c

^a*Alfred-Wegener-Institut, Helmholtz Zentrum für Polar- und Meeresforschung, Postfach 120161, 27515 Bremerhaven, Germany*

^b*Recherche en Prévision Numérique environnementale/Environnement Canada, 2121 route Transcanadienne, Dorval, Qc, Canada H9P 1J3*

^c*Universität Oldenburg, Ammerländer Heerstr. 114–118, 26129 Oldenburg*

Abstract

The most common representation of sea ice dynamics in climate models assumes that sea ice is a quasi-continuous non-normal fluid with a viscous-plastic rheology. This rheology leads to non-linear sea ice momentum equations that are notoriously difficult to solve. Recently a Jacobian-free Newton-Krylov (JFNK) solver was shown to solve the equations accurately at moderate costs. This solver is extended for massive parallel architectures and vector computers and implemented in a coupled sea ice-ocean general circulation model for climate studies. Numerical performance is discussed along with numerical difficulties in realistic applications with up to 1920 CPUs. The parallel JFNK-solver's scalability competes with traditional solvers although the collective communication overhead starts to show a little earlier. When accuracy of the solution is required (i.e. reduction of the residual norm of the momentum equations of more than one or two orders of magnitude) the JFNK-solver is unrivalled in efficiency. The new implementation opens up the opportunity to explore physical mechanisms in the context of large scale sea ice models and climate models and to clearly differentiate these physical effects from numerical artifacts.

Keywords: sea ice dynamics, numerical sea ice modeling, Jacobian-free Newton-Krylov solver, preconditioning, parallel implementation, vector implementation

*corresponding author

Email address: Martin.Losch@awi.de (Martin Losch)

1. Introduction

The polar oceans are geographically small compared to the world ocean, but still they are a very influential part of Earth's climate. Sea ice is an important component of the polar oceans. It acts as an insulator of heat and surface stress and without it atmospheric temperatures and hence flow patterns would be entirely different than today. Consequently, predicting future climate states or hindcasting previous ones requires accurate sea ice models [1, 2]. The motion of sea ice from formation sites to melting sites determines many aspects of the sea ice distribution and virtually all state-of-the-art sea ice models explicitly include a dynamics module.

Unfortunately, climate sea ice models necessarily contain many approximations that preclude the accurate description of sea ice dynamics. First of all, sea ice is usually treated as a quasi-continuous non-Newtonian fluid with a viscous-plastic rheology [3]. The assumption of quasi-continuity may be appropriate at low resolution but at high resolution (i.e. with a grid spacing on the order of kilometers) the scale of individual floes is reached and entirely new approaches may be necessary [4, 5, 6]. If continuity is acceptable (as in climate models with grid resolutions of tens of kilometers), the details of the rheology require attention [7, 8, 6]. Lemieux and Tremblay [9] and Lemieux et al. [10] demonstrated that the implicit numerical solvers that are used in climate sea ice models do not yield accurate solutions. These Picard solvers suffer from poor convergence rates so that iterating them to convergence is prohibitive [10]. Instead, a typical iterative process is terminated after a few (order two to ten) non-linear (or outer loop, OL) steps assuming falsely that the solution is sufficiently accurate [11, 9]. Without sufficient solution accuracy, the physical effects, that is, details of the rheology and improvements by new rheologies cannot be separated from numerical errors [12, 13]. Explicit methods may not converge at all [10].

Lemieux et al. [14] implemented a non-linear Jacobian-free Newton-Krylov (JFNK) solver in a serial sea model and demonstrated that this solver can give very accurate solutions compared to traditional solvers with comparatively low cost [10]. Here, we introduce and present the first JFNK-based sea ice model coupled to a general circulation model for parallel and vector computers. For this purpose, the JFNK solver was parallelized and vectorized. The parallelization required introducing a restricted additive Schwarz

36 method (RASM) [15] into the iterative preconditioning technique (line suc-
37 cessive relaxation, LSR) and the parallelization of the linear solver; the vector
38 code also required revisiting the convergence of the iterative preconditioning
39 method (LSR). The JFNK solver is matrix free, that is, only the product of
40 the Jacobian times a vector is required. The accuracy of this operation is
41 studied. Exact solutions with a tangent-linear model are compared to more
42 efficient finite-difference approaches.

43 Previous parallel JFNK solutions addressed compressible flow [16] or ra-
44 diative transfer problems [17]. The sea ice momentum equations stand apart
45 because the poor condition number of the coefficient matrix makes the sys-
46 tem of equations very difficult to solve [9]. The coefficients vary over many
47 orders of magnitude because they depend exponentially on the partial ice
48 cover within a grid cell (maybe comparable to Richards' equations for fluid
49 flow in partially saturated porous media [18]) and are a complicated function
50 (inverse of a square root of a quadratic expression) of the horizontal deriva-
51 tives of the solution, that is, the ice drift velocities. These are very different
52 in convergent motion where sea ice can resist large compressive stress and in
53 divergent motion where sea ice has very little tensile strength. As a conse-
54 quence, a successful JFNK solver for sea ice momentum equations requires
55 great care, and many details affect the convergence. For example, in contrast
56 to Godoy and Liu [17], we never observed convergence in realistic simulations
57 without sufficient preconditioning.

58 The paper is organized as follows. In Section 2 we review the sea ice mo-
59 mentum equations and the JFNK-solver; we describe the issues that needed
60 addressing and the experiments that are used to illustrate the performance
61 of the JFNK-solver. Section 3 discusses the results of the experiments and
62 conclusions are drawn in Section 4.

63 **2. Model and Methods**

64 For all computations we use the Massachusetts Institute of Technology
65 general circulation model (MITgcm) code [19, 20]. This code is a general
66 purpose, finite-volume algorithm on regular orthogonal curvilinear grids that
67 solves the Boussinesq and hydrostatic form of the Navier-Stokes equations
68 for an incompressible fluid with parameterizations appropriate for oceanic or
69 atmospheric flow. (Relaxing the Boussinesq and hydrostatic approximations
70 is possible, but not relevant here.) For on-line documentation of the general
71 algorithm and access to the code, see <http://mitgcm.org>. The MITgcm

72 contains a sea ice module whose dynamic part is based on Hibler’s [3] original
 73 work; the code has been rewritten for an Arakawa C-grid and extended to
 74 include different solution techniques and rheologies on curvilinear grids [12].
 75 The sea ice module serves as the basis for implementation of the JFNK solver.

76 *2.1. Model Equations and Solution Techniques*

77 The sea ice module of the MITgcm is described in Losch et al. [12]. Here
 78 we reproduce a few relevant aspects. The momentum equations are

$$79 \quad m \frac{D\mathbf{u}}{Dt} = -mf\mathbf{k} \times \mathbf{u} + \boldsymbol{\tau}_{air} + \boldsymbol{\tau}_{ocean} - m\nabla\phi(0) + \mathbf{F}, \quad (1)$$

80 where m is the combined mass of ice and snow per unit area; $\mathbf{u} = u\mathbf{i} + v\mathbf{j}$
 81 is the ice velocity vector; \mathbf{i} , \mathbf{j} , and \mathbf{k} are unit vectors in the x -, y -, and z -
 82 directions; f is the Coriolis parameter; $\boldsymbol{\tau}_{air}$ and $\boldsymbol{\tau}_{ocean}$ are the atmosphere-ice
 83 and ice-ocean stresses; $\nabla\phi(0)$ is the gradient of the sea surface height times
 84 gravity; and $\mathbf{F} = \nabla \cdot \boldsymbol{\sigma}$ is the divergence of the internal ice stress tensor σ_{ij} .
 85 Advection of sea-ice momentum is neglected. The ice velocities are used to
 86 advect ice compactness (concentration) c and ice volume, expressed as cell
 87 averaged thickness hc ; h is the ice thickness. The numerical advection scheme
 88 is a so-called 3rd-order direct-space-time method [21] with a flux limiter [22]
 89 to avoid unphysical over and undershoots. The remainder of the section
 90 focuses on solving (1).

91 For an isotropic system the stress tensor σ_{ij} ($i, j = 1, 2$) can be related to
 92 the ice strain rate tensor

$$93 \quad \dot{\epsilon}_{ij} = \frac{1}{2} \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right)$$

94

95 and the ice pressure

$$96 \quad P = P^* c h e^{[-C \cdot (1-c)]}$$

97

98 by a nonlinear viscous-plastic (VP) constitutive law [3, 11]:

$$99 \quad \sigma_{ij} = 2\eta\dot{\epsilon}_{ij} + [\zeta - \eta]\dot{\epsilon}_{kk}\delta_{ij} - \frac{P}{2}\delta_{ij}. \quad (2)$$

100 The ice pressure P , a measure of ice strength, depends on both thickness
 101 h and compactness (concentration) c ; the remaining constants are set to

102 $P^* = 27\,500 \text{ N m}^{-2}$ and $C = 20$. We introduce the shear deformation $\dot{\epsilon}_s =$
103 $\sqrt{(\dot{\epsilon}_{11} - \dot{\epsilon}_{22})^2 + \dot{\epsilon}_{12}^2}$, the shear divergence $\dot{\epsilon}_d = \dot{\epsilon}_{11} + \dot{\epsilon}_{22}$, and the abbreviation
104 $\Delta = \sqrt{\dot{\epsilon}_d^2 + \dot{\epsilon}_s^2/e^2}$. The nonlinear bulk and shear viscosities $\zeta = P/(2\Delta)$ and
105 $\eta = \zeta/e^2$ are functions of ice strain rate invariants and ice strength such that
106 the principal components of the stress lie on an elliptical yield curve with the
107 ratio of major to minor axis $e = 2$.

108 Substituting (2) into (1) yields equations in u and v that contain highly
109 non-linear viscosity-like terms with spatially and temporally variable coeffi-
110 cients ζ and η ; these terms dominate the momentum balance. Δ can be very
111 small where ice is thick and rigid so that ζ and η can span several orders of
112 magnitude making the non-linear equations very difficult to solve, and some
113 regularization is required. Following Lemieux et al. [10], the bulk viscosities
114 are bounded smoothly from above by imposing a maximum $\zeta_{\max} = P/(2\Delta^*)$,
115 with $\Delta^* = 2 \times 10^{-9} \text{ s}^{-1}$:

$$\begin{aligned}
\zeta &= \zeta_{\max} \tanh\left(\frac{P}{2 \min(\Delta, \Delta_{\min}) \zeta_{\max}}\right) \\
&= \frac{P}{2\Delta^*} \tanh\left(\frac{\Delta^*}{\min(\Delta, \Delta_{\min})}\right)
\end{aligned}
\tag{3}$$

117 $\Delta_{\min} = 10^{-20} \text{ s}^{-1}$ is chosen to avoid divisions by zero. Alternatively, one
118 could use a differentiable formula such as $\zeta = P/[2(\Delta + \Delta^*)]$, but in any case
119 the problem remains poorly conditioned. After regularizing the viscosities,
120 the pressure replacement method is used to compute the pressure as $2\Delta\zeta$
121 [23]. For details of the spatial discretization, see Losch et al. [12]. For the
122 following discussion, the temporal discretization is implicit in time following
123 Lemieux et al. [10].

124 The discretized momentum equations can be written in matrix notation
125 as

$$\mathbf{A}(\mathbf{x}) \mathbf{x} = \mathbf{b}(\mathbf{x}).
\tag{4}$$

126
127 The solution vector \mathbf{x} consists of the two velocity components u and v that
128 contain the velocity variables at all grid points and at one time level. In
129 the sea ice component of the MITgcm, as in many sea ice models, Eq. (4)
130 is solved with an iterative Picard solver: in the k -th iteration a linearized
131 form $\mathbf{A}(\mathbf{x}^{k-1}) \mathbf{x}^k = \mathbf{b}(\mathbf{x}^{k-1})$ is solved (in the case of the MITgcm it is an
132 LSR-algorithm [11], but other methods may be more efficient [24]). Picard
133 solvers converge slowly, but generally the iteration is terminated after only a
134 few non-linear steps [11, 9] and the calculation continues with the next time

135 level. Alternatively, the viscous-plastic constitutive law can be modified to
 136 elastic-viscous-plastic (EVP) to allow a completely explicit time stepping
 137 scheme [25]. These EVP-solvers are very popular because they are fast and
 138 efficient for massive parallel applications, but their convergence properties
 139 are under debate [10]. The EVP-solver in the MITgcm [12, 13] is extended
 140 to the modified EVP*-solver [10] for all EVP simulations.

141 The Newton method transforms minimizing the residual $\mathbf{F}(\mathbf{x}) = \mathbf{A}(\mathbf{x}) \mathbf{x} -$
 142 $\mathbf{b}(\mathbf{x})$ to finding the roots of a multivariate Taylor expansion of the residual
 143 \mathbf{F} around the previous ($k - 1$) estimate \mathbf{x}^{k-1} :

$$144 \quad \mathbf{F}(\mathbf{x}^{k-1} + \delta\mathbf{x}^k) = \mathbf{F}(\mathbf{x}^{k-1}) + \mathbf{F}'(\mathbf{x}^{k-1}) \delta\mathbf{x}^k \quad (5)$$

145 with the Jacobian $\mathbf{J} \equiv \mathbf{F}'$. The root $\mathbf{F}(\mathbf{x}^{k-1} + \delta\mathbf{x}^k) = 0$ is found by solving

$$146 \quad \mathbf{J}(\mathbf{x}^{k-1}) \delta\mathbf{x}^k = -\mathbf{F}(\mathbf{x}^{k-1}) \quad (6)$$

147 for $\delta\mathbf{x}^k$. The next (k -th) estimate is given by $\mathbf{x}^k = \mathbf{x}^{k-1} + a \delta\mathbf{x}^k$. In order
 148 to avoid overshoots the factor a is iteratively reduced in a line search ($a =$
 149 $1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \dots$) until $\|\mathbf{F}(\mathbf{x}^k)\| < \|\mathbf{F}(\mathbf{x}^{k-1})\|$, where $\|\cdot\| = \int \cdot dx^2$ is the L_2 -norm.
 150 In practice, the line search is stopped at $a = \frac{1}{8}$.

151 Forming the Jacobian \mathbf{J} explicitly is often avoided as “too error prone
 152 and time consuming” [26]. Instead, Krylov methods only require the action
 153 of \mathbf{J} on an arbitrary vector \mathbf{w} and hence allow a matrix free algorithm for
 154 solving Eq. (6) [26]. The action of \mathbf{J} can be approximated by a first-order
 155 Taylor series expansion [26]:

$$156 \quad \mathbf{J}(\mathbf{x}^{k-1}) \mathbf{w} \approx \frac{\mathbf{F}(\mathbf{x}^{k-1} + \epsilon\mathbf{w}) - \mathbf{F}(\mathbf{x}^{k-1})}{\epsilon} \quad (7)$$

157 or computed exactly with the help of automatic differentiation (AD) tools
 158 [16]. Besides the finite-difference approach we use TAF ([http://www.fastopt.](http://www.fastopt.de)
 159 [de](http://www.fastopt.de)) or TAMC [27] to obtain the action of \mathbf{J} on a vector. The MITgcm is
 160 tailored to be used with these tools [28] so that obtaining the required code
 161 with the help of a tangent linear model is straightforward.

162 We use the Flexible Generalized Minimum RESidual method (FGMRES,
 163 [29]) with right-hand side preconditioning to solve Eq. (6) iteratively starting
 164 from a first guess of $\delta\mathbf{x}_0^k = 0$. For the preconditioning matrix \mathbf{P} we choose a
 165 simplified form of the system matrix $\mathbf{A}(\mathbf{x}^{k-1})$ [14] where \mathbf{x}^{k-1} is the estimate
 166 of the previous Newton step $k - 1$. The transformed equation (6) becomes

$$167 \quad \mathbf{J}(\mathbf{x}^{k-1}) \mathbf{P}^{-1} \delta\mathbf{z} = -\mathbf{F}(\mathbf{x}^{k-1}), \quad \text{with} \quad \delta\mathbf{z} = \mathbf{P} \delta\mathbf{x}^k. \quad (8)$$

168 The Krylov method iteratively improves the approximate solution to (8) in
169 subspace $(\mathbf{r}_0, \mathbf{J}\mathbf{P}^{-1}\mathbf{r}_0, (\mathbf{J}\mathbf{P}^{-1})^2\mathbf{r}_0, \dots, (\mathbf{J}\mathbf{P}^{-1})^m\mathbf{r}_0)$ with increasing m ; $\mathbf{r}_0 =$
170 $-\mathbf{F}(\mathbf{x}^{k-1}) - \mathbf{J}(\mathbf{x}^{k-1})\delta\mathbf{x}_0^k$ is the initial residual of (6); $\mathbf{r}_0 = -\mathbf{F}(\mathbf{x}^{k-1})$ with the
171 first guess $\delta\mathbf{x}_0^k = 0$. We allow a Krylov-subspace of dimension $m = 50$ and we
172 do not use restarts. The preconditioning operation involves applying \mathbf{P}^{-1} to
173 the basis vectors $\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_m$ of the Krylov subspace. This operation is
174 approximated by solving the linear system $\mathbf{P}\mathbf{w} = \mathbf{v}_i$. Because $\mathbf{P} \approx \mathbf{A}(\mathbf{x}^{k-1})$,
175 we can use the LSR-algorithm [11] already implemented in the Picard solver.
176 Each preconditioning operation uses a fixed number of 10 LSR-iterations
177 avoiding any termination criterion. More details can be found in [14].

178 The non-linear Newton iteration is terminated when the L_2 -norm of the
179 residual is reduced by γ_{nl} with respect to the initial norm: $\|\mathbf{F}(\mathbf{x}^k)\| <$
180 $\gamma_{\text{nl}}\|\mathbf{F}(\mathbf{x}^0)\|$. Within a non-linear iteration, the linear FGMRES solver is
181 terminated when the residual is smaller than $\gamma_k\|\mathbf{F}(\mathbf{x}^{k-1})\|$ where γ_k is deter-
182 mined by

$$183 \quad \gamma_k = \begin{cases} \gamma_0 & \text{for } \|\mathbf{F}(\mathbf{x}^{k-1})\| \geq r, \\ \max\left(\gamma_{\min}, \frac{\|\mathbf{F}(\mathbf{x}^{k-1})\|}{\|\mathbf{F}(\mathbf{x}^{k-2})\|}\right) & \text{for } \|\mathbf{F}(\mathbf{x}^{k-1})\| < r, \end{cases} \quad (9)$$

184 so that the linear tolerance parameter γ_k decreases with the non-linear New-
185 ton step as the non-linear solution is approached. This inexact Newton
186 method is generally more robust and computationally more efficient than
187 exact methods [30, 26]. We choose $\gamma_0 = 0.99$, $\gamma_{\min} = 0.1$, and $r = \frac{1}{2}\|\mathbf{F}(\mathbf{x}^0)\|$;
188 we allow up to 100 Newton steps and a Krylov subspace of dimension 50. For
189 our experiments we choose γ_{nl} so that the JFNK (nearly) reaches numerical
190 working precision.

191 2.2. Parallelization

192 For a parallel algorithm, three issues had to be addressed:

- 193 (1) scalar product for computing the L_2 -norm
- 194 (2) parallelization of the Jacobian times vector operation
- 195 (3) parallelization of the preconditioning operation

196 The MITgcm is MPI-parallelized with domain decomposition [20]. We can
197 use the MITgcm primitives for computing global sums to obtain the scalar
198 product for the L_2 -norm. The parallel Jacobian times vector operation and
199 the preconditioning technique require that all fields are available sufficiently

200 far into the computational overlaps. This can be accomplished by one ex-
201 change (filling of overlaps, again by MITgcm primitives) for each velocity
202 component before these operations. The remaining parallelization of the pre-
203 conditioning operation is simplified by using the existing LSR-algorithm in
204 the Picard solver. The convergence of the iterative preconditioning method,
205 and hence of FGMRES linear solver, was greatly improved by introducing
206 a restrictive additive Schwarz method (RASM): in each LSR-iteration a so-
207 lution is obtained on each sub-domain plus overlap and the global solution
208 is combined by disposing of the overlaps [15]. At the end of each LSR-
209 iteration, the overlaps are filled once per velocity component. A so-called
210 parallel Newton-Krylov-Schwarz solver has been described in different con-
211 texts [e.g., 31, 32].

212 2.3. Vectorization

213 The MITgcm dynamic kernel code vectorizes with vector operation ra-
214 tios of 99% and higher on an NEC SX-8R vector computer. The only part
215 of the code where the algorithm is modified for better vectorization on vec-
216 tor computers is the LSR-method. This method solves tridiagonal systems
217 with the Thomas algorithm [33] along lines of constant j (or i) for the u (or
218 v) components separately. The Thomas algorithm cannot be vectorized so
219 that, for better vector performance, the order of the spatial loops have been
220 exchanged. For example, the Thomas algorithm for the i -direction is applied
221 to each component of a vector in j with the effect that the solution for $j - 1$
222 is not available when the j -th line is solved; instead the values of the previous
223 LSR-iterate are used (see Figure 1). This turns out to slow down the con-
224 vergence of the LSR-preconditioner enough to inhibit the convergence of the
225 FGMRES solver in many cases (which in turn affects the convergence of the
226 JFNK solver). As a solution to this problem the vectorized j -loops with loop
227 increment one is split into two loops with loop increments of two (a black
228 and a white loop), so that in the second (white) loop the solution at j can be
229 computed with an updated solution of the black loop at $j - 1$ and $j + 1$. This
230 “zebra” or line-coloring-method [34] improves the convergence of the LSR-
231 preconditioner to the extent that the preconditioned FGMRES solver (and
232 consequently the JFNK solver) regains the convergence that is expected—at
233 the cost of halved vector lengths. The LSR-vector code in the Picard solver
234 also suffers from slower convergence than the scalar code but that is com-
235 pensated by more iterations to satisfy a convergence criterion, so that the
236 “zebra”-method does not lead to a substantial improvement.

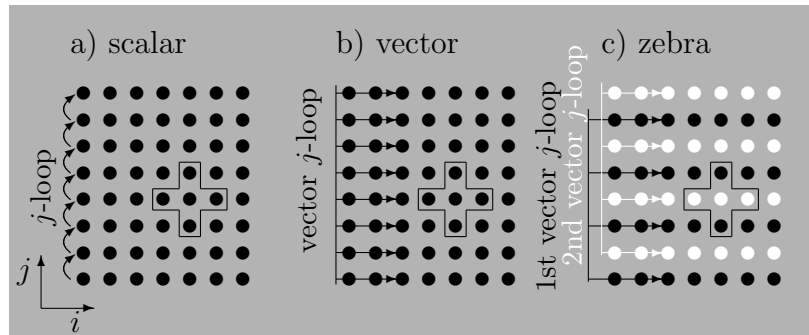


Figure 1: Schematic of LSR-algorithm for the u -component of the ice velocities: (a) the scalar code solves a tridiagonal system for each j -row sequentially, using known values of row $j - 1$ of the current sweep and values of row $j + 1$ of the previous sweep for the 5-point stencil (indicated by the cross); (b) the vector code solves all tridiagonal systems simultaneously, so that only information from previous sweep is available for $j \pm 1$; (c) the “zebra” code solves the black rows simultaneously and then in the second, white sweep the updated information of the black rows $j \pm 1$ can be used.

237 2.4. Experiments

238 We present simulations of two experimental configurations that demon-
 239 strate the overall performance of the JFNK with respect to parallel scaling
 240 and vectorization. Comparisons are made with the Picard solver and the
 241 EVP*-solver of the MITgcm sea ice module. Both configurations span the
 242 entire Arctic Ocean and in both cases the coupled sea ice-ocean system is
 243 driven by prescribed atmospheric reanalysis fields. The ice model is stepped
 244 with the same time step as the ocean model and both model components
 245 exchange fluxes of heat, fresh water, and momentum interfacial stress at
 246 each time level. The two configurations differ in resolution and integration
 247 periods. For practical reasons, the atmospheric boundary conditions (i.e.,
 248 the surface forcing data sets) are very different between these configurations,
 249 further excluding any direct comparisons between the simulations. The very
 250 interesting comparison of effects of resolution and solvers on climatically rel-
 251 evant properties of the solutions will be described elsewhere.

252 The first model is used for parallel scaling analysis only. It is based on a
 253 simulation with a 4 km grid spacing on an orthogonal curvilinear grid with
 254 1680 by 1536 grid points [35, 36]. Figure 2 shows the ice distribution and
 255 the shear deformation $\dot{\epsilon}_s$, both with many small scale structures and linear
 256 kinematic features (leads), on Dec-29-2006. For the scaling analysis, the sim-
 257 ulation is restarted in winter on Dec-29-2006 with a very small time step size

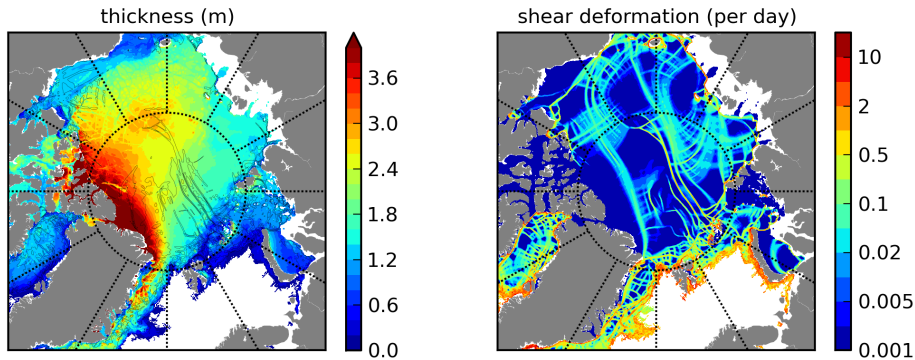


Figure 2: Thickness (m) and concentration (unlabeled contours of 90%, 95%, and 99%) and shear deformation (per day) of the 4 km resolution model on Dec-29-2006.

258 of 1 second for a few time steps. For long integrations this time step size is
 259 unacceptably small, but here it is necessary because at this resolution the
 260 system of equations is even more heterogeneous and ill-conditioned than at
 261 lower resolution and the convergence of JFNK (and other solvers) is slower
 262 [14]. With larger time steps the number of iterations to convergence (es-
 263 pecially when γ_{nl} is small) is different for different numbers of sub-domains
 264 (processors) and the scaling results are confounded. All simulations with this
 265 configuration are performed either on an SGI UV-100 cluster with Intel[®]
 266 Xeon[®] CPUs (E7-8837 @ 2.67 GHz) that is available at the computing center
 267 of the Alfred Wegener Institute (1–240 CPUs) or on clusters with Intel[®]
 268 Xeon[®] Gainestown processors (X5570 @ 2.93 GHz) (Nehalem EP) at the
 269 North German Supercomputing Alliance (Norddeutscher Verbund für Hoch-
 270 und Höchstleistungsrechnen, HLRN, <http://www.hlrn.de>) (8–1920 CPUs).
 271

272 The second model is run on 2 CPUs of an NEC SX-8R vector com-
 273 puter at the computing center of the Alfred Wegener Institute. For these
 274 simulations the Arctic Ocean is covered by a rotated quarter-degree grid
 275 along longitude and latitude lines so that the equator of the grid passes
 276 through the geographical North Pole and the grid spacing is approximately
 277 27 km; the time step size is 20 min. The model is started from rest with
 278 zero ice volume on Jan-01-1958 and integrated until Dec-31-2007 with inter-
 279 annually varying reanalysis forcing data of the CORE.v2 data set (<http://data1.gfdl.noaa.gov/nomads/forms/core/COREv2.html>). Model grid
 280 and configuration are similar to Karcher et al. [37]. Figure 3 shows the
 281

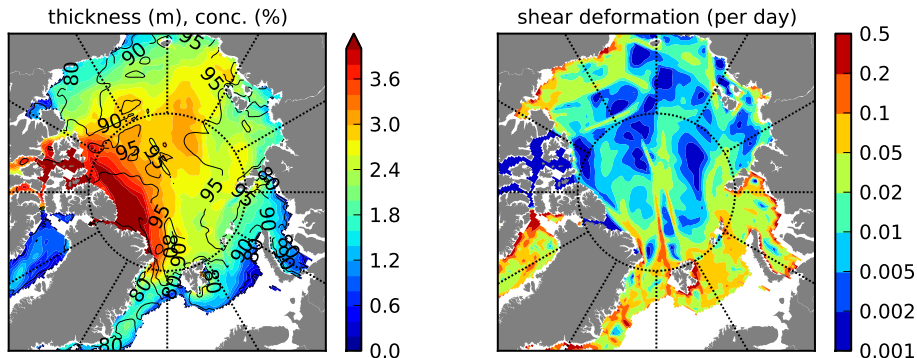


Figure 3: Example ice thickness, concentration (contours), and shear deformation (per day) of the coarse 27 km resolution model derived from velocity fields on Jun-30-1982.

282 thickness distribution and the shear deformation of Jun-30-1982 in the simu-
 283 lation. The ice fields are smooth compared to the 4 km-case (Figure 2), but
 284 some linear kinematic features also appear in the deformation fields. Note
 285 that over the 50 years of simulation (1,314,864 time levels) the JFNK-solver
 286 failed only 81 times to reach the convergence criterion of $\gamma_{nl} = 10^{-4}$ within
 287 100 iterations corresponding to a failure rate of 0.006%. To our knowledge
 288 this is the first successful coupled ice-ocean simulation with a JFNK-solver
 289 for the ice-dynamics.

290 3. Results

291 3.1. Effect of Jacobian times vector approximation

292 For this experiment, the coarse resolution simulation is restarted on Jan-
 293 01-1966 and the convergence criterion set to $\gamma_{nl} = 10^{-16}$ to force the solver
 294 to reach machine precision on the NEC SX-8R. Figure 4 shows that the
 295 convergence is a function of ϵ in (7), but the range of ϵ for which the finite-
 296 difference operation is sufficiently accurate is comfortably large. In practice,
 297 full convergence to machine precision will hardly be required so that we can
 298 give a range of $\epsilon \in [10^{-10}, 10^{-6}]$. In this case, using an exact Jacobian by
 299 AD only leads to a very small improvement of one Krylov iteration in the
 300 last Newton iteration before machine precision is reached. In all ensuing
 301 experiments we use the finite-difference approximation (7) with $\epsilon = 10^{-10}$.

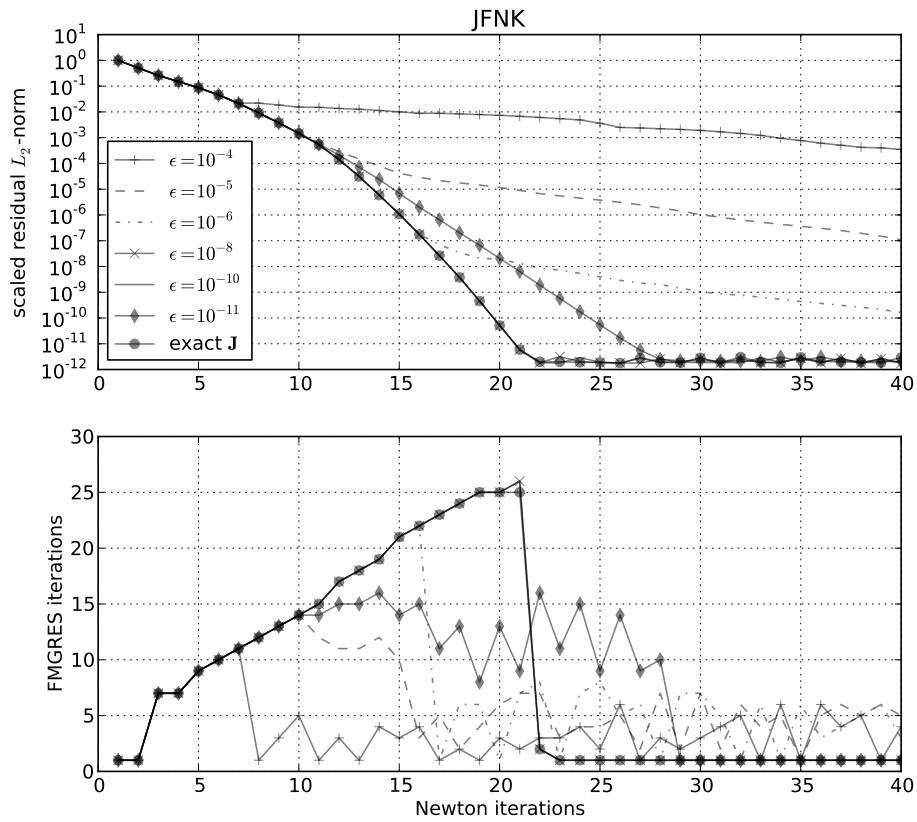


Figure 4: Convergence history of JFNK (top) and total number FGMRES iterations per Newton iteration (bottom) on the NEC SX-8R with different ϵ for the Jacobian times vector operation. The result with the exact Jacobian time vector operation by AD is also shown.

302 *3.2. Effect of zebra LSR-algorithm*

303 Figure 5 shows $\|\mathbf{F}(\mathbf{x}^k)\|$ as a function of the Newton iteration k for
304 three different treatments of the tridiagonal Thomas algorithm in the LSR-
305 preconditioner. The scalar code (Figure 1a) convergences very quickly, but
306 cannot be vectorized so that the time to solution is large. After exchanging
307 the i - and j -loops for better vector performance (Figure 1b), the good con-
308 vergence with the scalar code (solid line) is lost because the convergence rate
309 of the preconditioned FGMRES solver is lower (dashed line). Introducing
310 the “zebra”-method (Figure 1c) recovers the convergence completely (dash-
311 dotted line) and maintains the vector performance of the vector code with
312 low extra computational expenses; the code can be vectorized but vector
313 lengths are cut in half compared to the non-“zebra”-code.

314 *3.3. Effect of RASM on JFNK convergence*

315 Figure 6 shows that the convergence can be improved with a restricted
316 additive Schwarz Method (RASM) even with an overlap of only 1 grid point.
317 For an overlap of more than 1 the convergence can be still improved in
318 some cases, but not in all (not shown). In general, without writing special
319 exchange primitives for the sea-ice module, the size of the overlap is limited
320 to the total overlap required for general exchange MPI operation (usually not
321 greater than 5) minus the overlap required by the sea-ice dynamics solver (at
322 least 2).

323 *3.4. Parallel Scaling*

324 For a credible, unconfounded scaling analysis, the convergence history
325 of the JFNK-solver needs to be independent of the domain decomposition
326 (number of CPUs). For the following analysis the convergence history is ex-
327 actly the same for all domain decompositions until the 16th Newton iteration;
328 then the models start to deviate from each other because the summations in
329 the scalar product are performed in slightly different order with a different
330 number of sub-domains. As a consequence the number of Newton iterations
331 required to reach the convergence criterion of $\gamma_{nl} = 10^{-10}$ is also different
332 for different domain decompositions. This effect increases with larger time
333 steps. For the present case, the number of Newton and Krylov steps varies
334 moderately between simulations of 4 time steps (121–127 Newton steps and
335 714–754 Krylov steps), so that we can use the results for a scaling analysis.
336 For comparison, the number of LSR iterations in the Picard solver varies

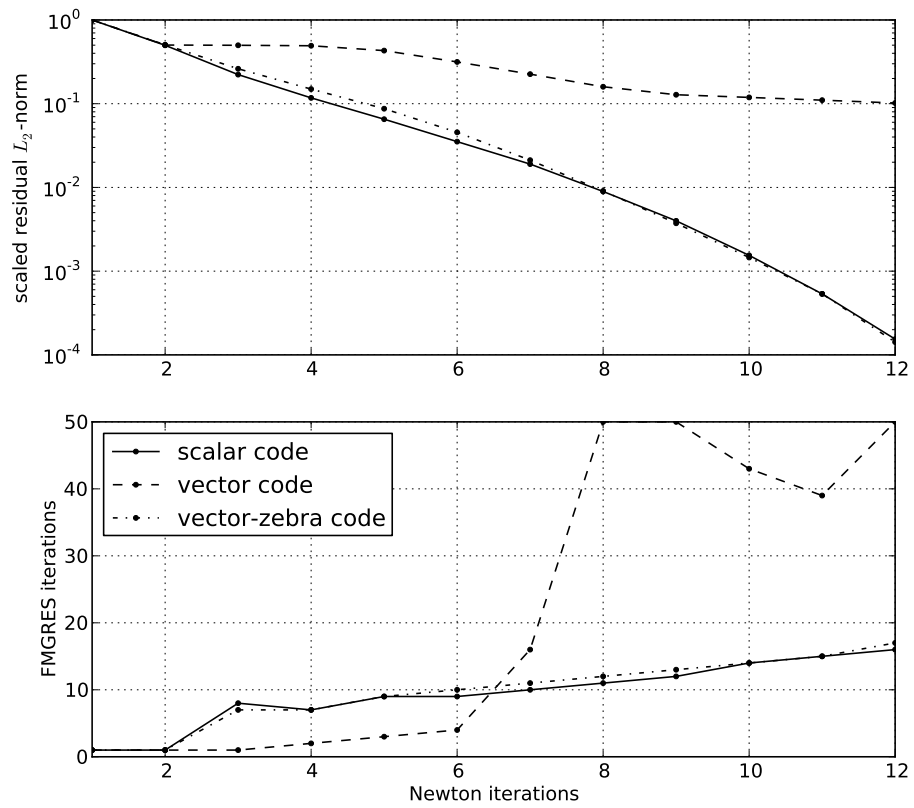


Figure 5: Convergence history of JFNK (top) and total number FGMRES iterations per Newton iteration (bottom) on the NEC SX-8R with different vectorization methods for the tridiagonal Thomas algorithm in LSR.

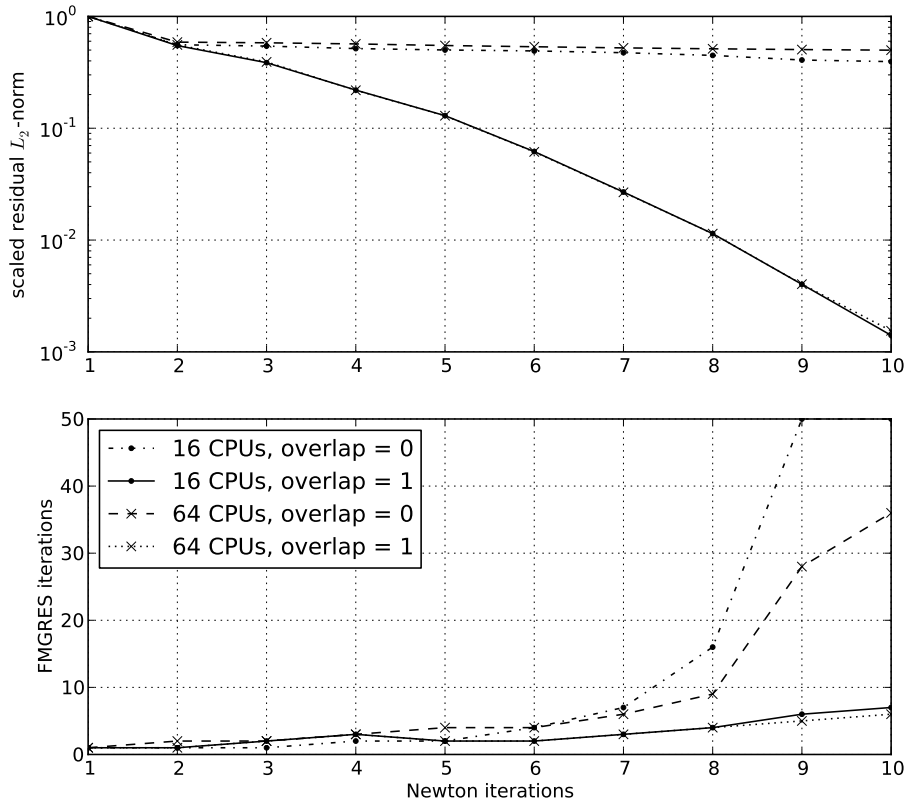


Figure 6: Convergence history of JFNK (top) and total number FGMRES iterations per Newton iteration (bottom) on the SGI-UV100 with and without RASM for 16 and 64 CPUs. “overlap = 0” refers to no overlap (no RASM) and “overlap = 1” to RASM with an overlap of one.

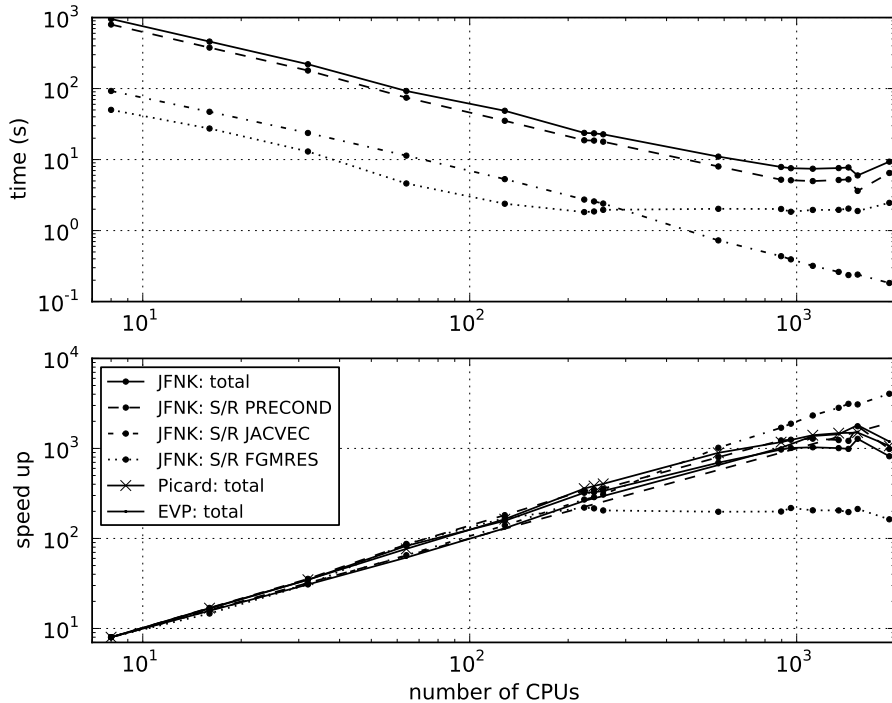


Figure 7: Time for four time steps (top) and relative speed up (bottom) as a function of number of CPUs for the 4 km resolution configuration on the HLRN computer. The absolute times for the EVP*- and Picard solver are not included.

337 between 3986 and 4020 in 4 time steps of the same configuration. We con-
 338 firmed that with a scalar product that preserves the order of summation, this
 339 dependence on domain decomposition can be eliminated completely at the
 340 cost of very inefficient code.

341 Figure 7 shows the scaling data obtained from running the models for
 342 4 time steps. For comparison, the results of the Picard solver and the
 343 EVP*-solver are included. The EVP*-solver only includes point-to-point
 344 communications, but the Picard solver requires point-to-point and collec-
 345 tive communications. The JFNK-solver scales linearly as the Picard and the
 346 EVP*-solver, but reaches a communication overhead earlier (at 10^3 CPUs).
 347 The routines responsible for this overhead are the many scalar products in
 348 the Krylov-method (S/R FGMRES) and the many point-to-point commu-
 349 nications within the preconditioning step (S/R PRECOND) (see also [38]).

350 Routines that do not require any communication (e.g, S/R JACVEC carries
 351 out the Jacobian times vector operation of Eq. (7)) scale linearly to the max-
 352 imum number of CPUs of 1920, after which the sub-domain size becomes
 353 too small to allow linear scaling for the ocean model. Note that both EVP*
 354 and Picard solver loose linear scalability above 10^3 CPUs indicating general
 355 limits of the system.

356 *3.5. Comparison of JFNK to Picard (LSR) and EVP* convergence history*

357 Figure 8 shows the convergence history of the Picard solver for different
 358 termination criteria of the linear LSR-solver and of the JFNK and EVP*
 359 solver as a function of scaled linear (inner) iterations. Results are obtained
 360 with the 27 km resolution on the NEC SX-8R. The linear iterations are scaled
 361 by the time to solution divided by the total number of linear iterations. For
 362 the EVP*-solver, the sub-cycling steps are strictly speaking non-linear iter-
 363 ations, but one such step costs approximately as much as one iteration of a
 364 linear solver so that they are only plotted with the linear iterations and not
 365 with the non-linear iterations. This pseudo-timing of the EVP*-solver may
 366 overestimate its actual cost relative to the other solvers, but in our case the
 367 EVP*-solver never converges anyway. For tighter termination criteria the
 368 non-linear convergence of the Picard solver improves per non-linear iteration
 369 as expected, but also the computational cost increases. Initially, the conver-
 370 gence is actually faster (assuming that each linear iteration takes the same
 371 time) for weaker linear convergence criteria. For the case of $\epsilon_{\text{LSR}} = 10^{-2}$, the
 372 Picard solver even outperforms the JFNK-solver for the first 0.1 s (approx-
 373 imately 50 linear iterations). Otherwise, the JFNK-solver is more efficient
 374 [14], especially after the first couple of non-linear steps. Hence we can con-
 375 firm that for smaller γ_{nl} the computational advantage of the JFNK-solver
 376 over the Picard-solver increases [14]. The EVP*-solver converges faster than
 377 the Picard solver for the first 0.5 s (approximately 250 iterations) and for
 378 LSR-convergence criteria $\epsilon_{\text{LSR}} < 10^{-4}$, but then it flattens out and oscillates
 379 while the Picard solver continues to reduce the residual. For LSR-convergence
 380 criteria $\epsilon_{\text{LSR}} \geq 10^{-4}$, the Picard solver always converges faster (see also [10]).

381
 382 Note that the usual representation of the residual L_2 -norm as a function of
 383 non-linear iterations (bottom panel of Figure 8) more clearly shows that the
 384 JFNK is always more efficient per non-linear iteration, but this representation
 385 is misleading if one is interested in the computational advantage of the JFNK
 386 solver. Here the upper panel gives a more realistic representation.

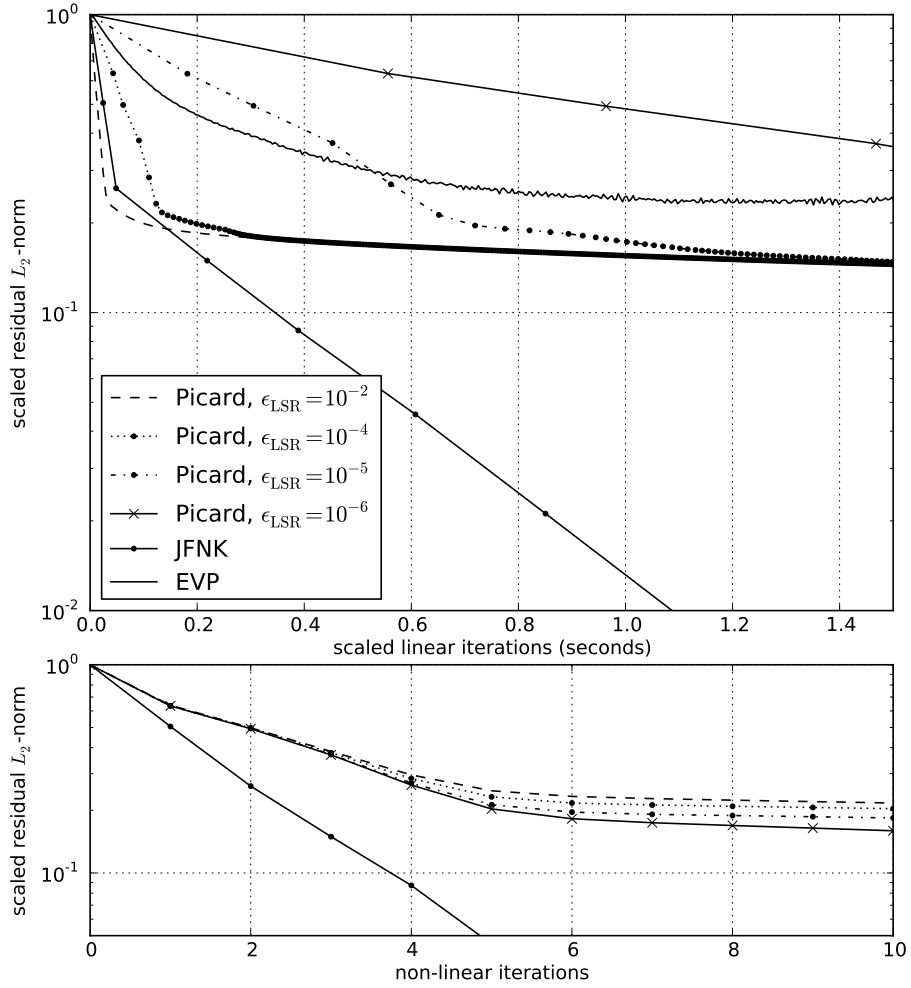


Figure 8: Convergence history of JFNK, EVP*, and Picard solver with different termination criteria for the linear LSR-solver as a function of the number of linear iterations on the NEC SX-8R (top). The number of linear iterations is scaled by the time to solution over total number of linear iteration. The dots and crosses mark the beginning of a new non-linear iteration. The bottom panel shows the residual (scaled by the initial residual) as a function of non-linear iterations.

387 4. Discussion and Conclusions

388 Applying the JFNK-method for solving the momentum equations in the
389 sea-ice module of a general circulation model for climate studies requires
390 adaptation and optimization of the method to high performance computer
391 environments. After parallelization and vectorization, the JFNK solver is
392 as successful as the serial version [14, 10] in minimizing the L_2 -norm of the
393 residual of the equations. In our experiments the ratio of computational effort
394 (measured in number of iterations of the linear solver) to achieved residual
395 reduction is better for the JFNK-solver than for the traditional Picard-solver
396 and the EVP*-solver. Only for very few linear iterations, a properly tuned
397 Picard-solver can outperform the JFNK-solver. A combination of Picard and
398 JFNK-solver may be an optimal solution [18].

399 The JFNK-solver runs efficiently on vector computers (here: NEC SX-
400 8R), and it scales on massive parallel computers down to a domain size of
401 approximately 50 by 50 grid points (approximately 1000 CPUs in our test).
402 The bottlenecks are a communication overhead in point-to-point exchanges of
403 the preconditioning operation and eventually a communication overhead in-
404 curred by many scalar products (collective communication) in the FGMRES-
405 solver. Alternative methods, for example, replacing the Gram-Schmidt-
406 orthogonalization in the FGMRES implementation by a Householder-reflection
407 method may alleviate the latter [17], but the former overhead will be felt by
408 all solvers. The flattening of the scaling curves of the Picard- and EVP*-
409 solver at the very end of the scaling curve is likely caused by the point-to-
410 point communication overhead.

411 While adapting the JFNK-solver to parallel or vector architectures is gener-
412 ally straightforward, the preconditioning operator requires more care. This
413 operation is the single most expensive routine in the JFNK-code (Figure 7),
414 because in each FGMRES-iteration it requires (in our case) ten LSR-loops,
415 each with one exchange of the solution vector field, so that an efficient treat-
416 ment of this part of the code is very important. Further, the convergence of
417 the FGMRES linear solver critically depends on the preconditioning opera-
418 tion and required introducing a restricted additive Schwarz Method (RASM)
419 with an overlap of at least one for parallel applications. For the vector code,
420 the LSR-preconditioner requires a “zebra”-method to ensure good perfor-
421 mance of the FGMRES solver. Without the RASM and “zebra” methods,
422 the preconditioned FGMRES sometimes does not converge before the allowed
423 maximum 50 Krylov iterations. These failures of FGMRES then affects the

424 nonlinear convergence of the JFNK solver. Furthermore, as our JFNK solver
425 is based on an inexact Newton method, a lower convergence rate of the
426 preconditioned FGMRES solver can also affect the overall nonlinear conver-
427 gence.

428 In order to reduce the computational cost of the expensive iterative LSR-
429 preconditioner, a direct (but approximate) procedure, such as a variant of
430 incomplete LU factorization (ILU), could be used. Such a direct method re-
431 quires only one set of point-to-point communications per FGMRES iteration
432 (instead of ten). Since the factorization itself is difficult to parallelize, the
433 method operates sequentially on each of the sub-domains defined by RASM.
434 There are methods for partial vectorization of ILU [39]. The approximate
435 nature of such a preconditioning operation may require more iterations of
436 FGMRES, and the actual overall performance remains to be demonstrated.

437 The accuracy of the Jacobian times vector operation was found to be less
438 critical. The exact operation with code obtained with AD slightly reduced
439 the number required iterations to reach work precision compared to forward
440 finite-difference (FD) code with a comfortable range of increments ϵ . With
441 the AD-code the JFNK-solver still took slightly more time (order 2%), be-
442 cause each Jacobian times vector operation requires two model evaluations,
443 forward model and tangent linear model, while the forward FD code requires
444 only one extra forward model evaluation. The AD-code evaluates forward
445 and tangent-linear model simultaneously, explaining the small overhead.

446 The current practice in climate modeling of using a Picard solver with
447 a low number of non-linear iterations or using the fast but poorly converg-
448 ing EVP-solver leads to approximate solutions (large residuals) of the sea
449 ice momentum equations. Investing extra computational time with a JFNK-
450 solver—for example, 500 LSR-iterations per time level instead of order 20—
451 can reduce this residual by 2 orders of magnitude and more. It has been
452 demonstrated that the differences between sea ice models with more and less
453 accurate solvers can easily reach 2–5 cm/s in ice drift velocities and 50 cm
454 to meters in ice thickness after only one month of integration [9]. These
455 differences are comparable to the differences due to other parameter choices
456 such as the advection scheme for thickness and concentration or the choice
457 of rheology, boundary conditions, or even grid-staggering [12]. We will not
458 speculate to what extent the extra accuracy of a JFNK-solver is required in
459 climate models, but for studying details of sea ice physics and rheology, an
460 accurate solver-technology seems in place to be able to differentiate between
461 numerical artifacts and physical effects. Our implementation of a parallel

462 JFNK-solver for sea ice dynamics in an ocean general circulation model is a
463 tool to explore new questions of rheology and sea-ice dynamics in the con-
464 text of large-scale and computationally challenging simulations that require
465 parallelized codes.

466 *Acknowledgements.* We thank Dimitris Menemenlis and An Nguyen for pro-
467 viding the high-resolution configuration to us. Some of the computations
468 were carried out at the North German Supercomputing Alliance (Nord-
469 deutscher Verbund für Hoch- und Höchstleistungsrechnen, HLRN).

470 **References**

- 471 [1] A. Proshutinsky, Z. Kowalik, Preface to special section on Arctic Ocean
472 Model Intercomparison Project (AOMIP) studies and results, *J. Geo-*
473 *phys. Res.* 112 (2007).
- 474 [2] P. Rampal, J. Weiss, C. Dubois, J.-M. Campin, IPCC climate models
475 do not capture Arctic sea ice drift acceleration: Consequences in terms
476 of projected sea ice thinning and decline, *J. Geophys. Res.* 116 (2011).
- 477 [3] W. D. Hibler, III, A dynamic thermodynamic sea ice model, *J. Phys.*
478 *Oceanogr.* 9 (1979) 815–846.
- 479 [4] A. V. Wilchinsky, D. L. Feltham, Modelling the rheology of sea ice as
480 a collection of diamond-shaped floes, *Journal of Non-Newtonian Fluid*
481 *Mechanics* 138 (2006) 22–32.
- 482 [5] D. L. Feltham, Sea ice rheology, *Ann. Rev. Fluid Mech.* 40 (2008)
483 91–112.
- 484 [6] M. Tsamados, D. L. Feltham, A. V. Wilchinsky, Impact of a new
485 anisotropic rheology on simulations of Arctic sea ice, *J. Geophys. Res.*
486 118 (2013) 91–107.
- 487 [7] K. Wang, C. Wang, Modeling linear kinematic features in pack ice, *J.*
488 *Geophys. Res.* 114 (2009).
- 489 [8] L. Girard, S. Bouillon, W. Jérôme, D. Amitrano, T. Fichefet, V. Legat,
490 A new modelling framework for sea ice models based on elasto-brittle
491 rheology, *Ann. Glaciol.* 52 (2011) 123–132.

- 492 [9] J.-F. Lemieux, B. Tremblay, Numerical convergence of viscous-plastic
493 sea ice models, *J. Geophys. Res.* 114 (2009).
- 494 [10] J.-F. Lemieux, D. Knoll, B. Tremblay, D. M. Holland, M. Losch, A
495 comparison of the Jacobian-free Newton-Krylov method and the EVP
496 model for solving the sea ice momentum equation with a viscous-plastic
497 formulation: a serial algorithm study, *J. Comp. Phys.* 231 (2012) 5926–
498 5944.
- 499 [11] J. Zhang, W. D. Hibler, III, On an efficient numerical method for mod-
500 eling sea ice dynamics, *J. Geophys. Res.* 102 (1997) 8691–8702.
- 501 [12] M. Losch, D. Menemenlis, J.-M. Campin, P. Heimbach, C. Hill, On
502 the formulation of sea-ice models. Part 1: Effects of different solver
503 implementations and parameterizations, *Ocean Modelling* 33 (2010)
504 129–144.
- 505 [13] M. Losch, S. Danilov, On solving the momentum equations of dynamic
506 sea ice models with implicit solvers and the Elastic Viscous-Plastic tech-
507 nique, *Ocean Modelling* 41 (2012) 42–52.
- 508 [14] J.-F. Lemieux, B. Tremblay, J. Sedláček, P. Tupper, S. Thomas,
509 D. Huard, J.-P. Auclair, Improving the numerical convergence of
510 viscous-plastic sea ice models with the Jacobian-free Newton-Krylov
511 method, *J. Comp. Phys.* 229 (2010) 2840–2852.
- 512 [15] X.-C. Cai, M. Sarkis, A restricted additive Schwarz preconditioner for
513 general sparse linear systems, *SIAM J. Sci. Comput.* 21 (1999) 792–797.
- 514 [16] P. D. Hovland, L. C. McInnes, Parallel simulation of compressible
515 flow using automatic differentiation and PETSc, *Parallel Computing*
516 27 (2001) 503–519.
- 517 [17] W. F. Godoy, X. Liu, Parallel Jacobian-free Newton Krylov solution of
518 the discrete ordinates method with flux limiters for 3D radiative transfer,
519 *J. Comp. Phys.* 231 (2012) 4257–4278.
- 520 [18] C. Paniconi, M. Putti, A comparison of Picard and Newton iteration
521 in the numerical solution of multidimensional variably saturated flow
522 problems, *Water Resour. Res.* 30 (1994) 3357–3374.

- 523 [19] J. Marshall, A. Adcroft, C. Hill, L. Perelman, C. Heisey, A finite-volume,
524 incompressible Navier Stokes model for studies of the ocean on parallel
525 computers, *J. Geophys. Res.* 102 (1997) 5753–5766.
- 526 [20] MITgcm Group, MITgcm User Manual, Online documentation,
527 MIT/EAPS, Cambridge, MA 02139, USA, 2012. [http://mitgcm.org/
528 public/r2_manual/latest/online_documents](http://mitgcm.org/public/r2_manual/latest/online_documents).
- 529 [21] W. Hundsdorfer, R. A. Trompert, Method of lines and direct discretiza-
530 tion: a comparison for linear advection, *Applied Numerical Mathematics*
531 13 (1994) 469–490.
- 532 [22] P. Roe, Some contributions to the modelling of discontinuous flows,
533 in: B. Engquist, S. Osher, R. Somerville (Eds.), *Large-Scale Computa-
534 tions in Fluid Mechanics*, volume 22 of *Lectures in Applied Mathematics*,
535 American Mathematical Society, 1985, pp. 163–193.
- 536 [23] C. A. Geiger, W. D. Hibler, III, S. F. Ackley, Large-scale sea ice drift
537 and deformation: Comparison between models and observations in the
538 western Weddell Sea during 1992, *J. Geophys. Res.* 103 (1998) 21893–
539 21913.
- 540 [24] J.-F. Lemieux, B. Tremblay, S. Thomas, J. Sedláček, L. A. Mysak, Using
541 the preconditioned Generalized Minimum RESidual (GMRES) method
542 to solve the sea-ice momentum equation, *J. Geophys. Res.* 113 (2008).
- 543 [25] E. C. Hunke, J. K. Dukowicz, The elastic-viscous-plastic sea ice dynam-
544 ics model in general orthogonal curvilinear coordinates on a sphere—
545 incorporation of metric terms, *Mon. Weather Rev.* 130 (2002) 1847–
546 1865.
- 547 [26] D. Knoll, D. Keyes, Jacobian-free Newton-Krylov methods: a survey of
548 approaches and applications, *J. Comp. Phys.* 193 (2004) 357–397.
- 549 [27] R. Giering, T. Kaminski, Recipes for adjoint code construction, *ACM*
550 *Trans. Math. Softw.* 24 (1998) 437–474.
- 551 [28] P. Heimbach, C. Hill, R. Giering, An efficient exact adjoint of the parallel
552 MIT general circulation model, generated via automatic differentiation,
553 *Future Generation Computer Systems (FGCS)* 21 (2005) 1356–1371.

- 554 [29] Y. Saad, A flexible inner-outer preconditioned GMRES method, SIAM
555 J. Sci. Comput. 14 (1993) 461–469.
- 556 [30] S. C. Eisenstat, H. F. Walker, Choosing the forcing terms in an inexact
557 Newton method, SIAM J. Sci. Comput. 17 (1996) 16–32.
- 558 [31] X.-C. Cai, W. D. Gropp, D. E. Keyes, M. D. Tidriri, Newton-Krylov-
559 Schwarz methods in CFD, in: F.-K. Hebeker, R. Rannacher, G. Wittum
560 (Eds.), Proceedings of the International Workshop on the Navier-Stokes
561 Equations, pp. 17–30.
- 562 [32] X.-C. Cai, W. D. Gropp, D. E. Keyes, R. G. Melvin, D. P. Young, Par-
563 allel Newton-Krylov-Schwarz algorithms for the transonic full potential
564 equation, SIAM J. Sci. Comput. 19 (1998) 246–265.
- 565 [33] L. H. Thomas, Elliptic problems in linear differential equations over a
566 network, Watson Sci. Comput. Lab Report, Columbia University, New
567 York, 1949.
- 568 [34] I. S. Duff, G. A. Meurant, The effect of ordering on preconditioned
569 conjugate gradients, BIT Numerical Mathematics 29 (1989) 635–657.
- 570 [35] A. T. Nguyen, R. Kwok, D. Menemenlis, Source and pathway of the
571 Western Arctic upper halocline in a data-constrained coupled ocean and
572 sea ice model, J. Phys. Oceanogr. 43 (2012) 80–823.
- 573 [36] E. Rignot, I. Fenty, D. Menemenlis, Y. Xu, Spreading of warm ocean
574 waters around Greenland as a possible cause for glacier acceleration,
575 Ann. Glaciol. 53 (2012) 257–266.
- 576 [37] M. Karcher, A. Beszczynska-Möller, F. Kauker, R. Gerdes, S. Heyen,
577 B. Rudels, U. Schauer, Arctic ocean warming and its consequences for
578 the Denmark Strait overflow, J. Geophys. Res. 116 (2011).
- 579 [38] C. Hill, D. Menemenlis, B. Ciotti, C. Henze, Investigating solution
580 convergence in a global ocean model using a 2048-processor cluster of
581 distributed shared memory machines, Scientific Programming 12 (2007)
582 107–115.
- 583 [39] J. J. F. M. Schlichting, H. A. van der Vorst, Solving 3D block bidiago-
584 nal linear systems on vector computers, Journal of Computational and
585 Applied Mathematics 27 (1989) 323–330.