

34 *Key words:* ice sheet model, ice rheology, Newton-Krylov, GMRES, ILU

35 **1. Introduction**

36 During the past decade, there have been major changes on both the
37 Greenland and Antarctic ice sheets as a result of ice dynamics. In Green-
38 land, many outlet glaciers underwent acceleration, thinning and retreat [1,
39 2, 3, 4, 5, 6] with a consequent increased contribution to global sea level.
40 In Antarctica, land-based glaciers flowing into the Larsen A and Larsen B
41 ice shelves sped up after those ice shelves collapsed [7, 8], and the Pine Is-
42 land and Thwaites glaciers continued to accelerate and thin [9, 10]. For both
43 Greenland and Antarctica, these dynamical changes are largely attributable
44 to atmospheric and oceanic forcing [6, 11, 12, 13].

45
46 For 2000-2008, one half of the total sea level rise from Greenland can be
47 attributed to ice dynamics [14]. In Antarctica, recent analytical and numer-
48 ical modeling studies confirm the potential for large-scale, dynamical insta-
49 bility, which could increase Antarctica's contribution to sea level by orders
50 of magnitude [15, 16, 17]. The overall consensus in the scientific literature
51 from the past decade is that Earth's large ice sheets respond to external cli-
52 mate forcing much faster than was previously thought possible and that ice
53 dynamics plays an important role in controlling the rate of mass loss to the
54 oceans [18, 19, 20].

55
56 Because our understanding of ice sheet dynamics and the controlling phys-
57 ical processes is limited, it remains difficult to make plausible estimates for
58 the magnitude of sea level rise associated with future changes in the Green-
59 land and Antarctic ice sheets. In its Fourth Assessment Report (AR4), the
60 Intergovernmental Panel on Climate Change [21] failed to provide a best esti-
61 mate or even an upper bound for future sea level rise from ice sheets, largely
62 due to this limited understanding. Our current inability to predict the fu-
63 ture evolution of ice sheets is demonstrated by the fact that most existing
64 ice sheet models fail to mimic or provide insight into the observed, dramatic
65 changes occurring on ice sheets.

66
67 One reason for this failing is the prevalence of models based on the "Shal-
68 low Ice Approximation" (SIA) for ice dynamics (e.g. [22]), which assumes

69 that all of the geometric driving stress is balanced locally through vertical
70 shearing in the ice column [23]. While the SIA greatly simplifies the numerical
71 solution of the momentum and mass conservation equations, the underlying
72 assumptions do not hold for the regions of the ice sheet that control the ma-
73 jority of the mass flux to the oceans (e.g. outlet glaciers, ice streams, and ice
74 shelves). To accurately simulate the flow in these regions, a more complete
75 description of the momentum balance is required, for example that given by
76 solving the Stokes flow [24] or first-order equations [25, 26, 27]. However,
77 the numerical implementation of these "higher order" momentum approxi-
78 mations is much more challenging than for the SIA approximation.

79
80 In this work, we focus on improving the numerical performance of ice
81 dynamics represented in the Community Ice Sheet Model, Glimmer-CISM.
82 The original model [22] was based on the SIA, but a first-order momen-
83 tum balance has recently been added [28]. Unlike the SIA, the first-order
84 formulation also accounts for the effects of longitudinal and lateral stress
85 gradients and, assuming appropriate boundary conditions, is able to repre-
86 sent the full continuum of ice flow observed on ice sheets, from relatively
87 slow inland flow to relatively fast ice stream and ice shelf flow. The model
88 participated in the higher-order model inter-comparison benchmarking study
89 (ISMIP-HOM) and, for tests A-E, all outputs were within one standard devi-
90 ation of the mean defined by other models of its type (additional information
91 on the results of the benchmarking study can be found in [29]).

92
93 The existing solution to the first-order momentum equations in Glimmer-
94 CISM involves (1) a splitting of the momentum equation into its u and v
95 components (x and y directions in map view) [25], (2) solution of the linearized
96 system for v by moving the u terms to the right-hand side and treating them
97 as known source terms (and vice versa), (3) a Picard iteration to handle
98 the nonlinearity associated with the ice rheology (as discussed below), and
99 (4) when appropriate, a correction to the solution from the Picard iteration
100 using the "unstable manifold correction" scheme of [30]. However, as has
101 been shown in previous work [31, 32], a Picard treatment of the nonlinearity
102 generally leads to undesirably slow rates of convergence.

103
104 This paper describes the implementation of a more computationally ef-
105 ficient nonlinear solver, the Jacobian-Free Newton-Krylov (JFNK) method,
106 into Glimmer-CISM. The JFNK method has many advantages: the rate of

107 convergence can be nearly quadratic in the vicinity of the solution, it is scal-
108 able (if care is taken with the preconditioning operator), and the Jacobian
109 does not need to be explicitly formed and stored. This last point is especially
110 important; for complicated problems, forming the Jacobian is a difficult de-
111 velopment task. While the JFNK method has been applied successfully to
112 other fields in the Earth sciences (e.g., [33, 34, 35]), to our knowledge this is
113 the first time it has been applied to ice sheet modeling.

114

115 Our JFNK implementation re-uses much of the code from the existing
116 nonlinear solver. This approach has two clear advantages. First, because it
117 takes advantage of existing code it is easy to implement, and second, the ma-
118 jority of that code has already been extensively tested. Following this, the
119 preconditioning operator is derived directly from the existing linear solver
120 of the Picard scheme, an approach sometimes referred to as physics based
121 preconditioning (e.g., [33]).

122

123 The next section (section 2) gives an overview of the first-order momen-
124 tum equation and the related boundary conditions that govern the ice dy-
125 namics in Glimmer-CISM. Section 3 describes the standard Picard solver and
126 the new JFNK solver in detail. In section 4, we describe a number of test
127 cases used to compare the Picard and JFNK solvers. The results of that com-
128 parison, the computational efficiency and robustness of Picard versus JFNK,
129 are given in section 5. Finally, in section 6 we summarize, make concluding
130 remarks, and discuss ongoing work to further improve the numerical perfor-
131 mance of Glimmer-CISM.

132

133 **2. Ice sheet momentum equation with a first-order formulation**

134 A complete description of the ice-sheet-evolution problem requires so-
135 lution of the relevant mass, momentum, and energy conservation equations.
136 Our numerical implementation involves a splitting in time such that ice thick-
137 ness and temperature are treated explicitly in the momentum equation. Here,
138 we focus only on the efficient solution of the momentum balance equation.
139 Consistent with an incompressible viscous fluid in a low Reynolds number
140 flow, the inertial and advective terms on the left-hand side of the momentum
141 equations are ignored, (similarly, the Coriolis term is neglected).

142

143 A brief description of the first-order momentum equations and boundary
 144 conditions is given here. For more details, the reader is referred to [28] and
 145 [25]. More discussion on the derivation of the first-order equations (starting
 146 from the full Stokes equations) and their formal accuracy can be found in
 147 [26, 27]. In the discussion below we assume a right-handed Cartesian coor-
 148 dinate system with x , y , and z representing the two horizontal and vertical
 149 directions, respectively.

150

151 Consistent with a first-order scaling of the full Stokes equations [26, 27],
 152 the vertical normal stress is balanced by the hydrostatic pressure. This sim-
 153 plification, along with incompressibility and some rearranging reduces the
 154 full Stokes equations with four unknowns (the three velocity components u ,
 155 v and w and the pressure) to two equations for the two horizontal velocity
 156 components, u and v ,

$$\frac{\partial}{\partial x} (2\sigma'_{xx} + \sigma'_{yy}) + \frac{\partial \sigma'_{xy}}{\partial y} + \frac{\partial \sigma'_{xz}}{\partial y} = \rho g \frac{\partial s}{\partial x} \quad (1)$$

$$\frac{\partial}{\partial x} (2\sigma'_{yy} + \sigma'_{xx}) + \frac{\partial \sigma'_{xy}}{\partial y} + \frac{\partial \sigma'_{yz}}{\partial y} = \rho g \frac{\partial s}{\partial y} \quad (2)$$

157 where the σ'_{ij} are the deviatoric stresses, ρ is the ice density, g is the gravita-
 158 tional acceleration and $z = s(x, y)$ defines the upper surface of the ice sheet.
 159 The vertical component of velocity, w , can be recovered from the solutions
 160 for u and v through incompressibility.

161

162 The right-hand side (RHS) terms in equations (1) and (2) represent the
 163 volumetric body forces. For the constitutive equation (i.e. the relation be-
 164 tween applied stresses and resulting deformations) we use Glen's flow law
 165 [23] in a Newtonian form:

$$\sigma'_{ij} = 2\eta \dot{\epsilon}_{ij} \quad (3)$$

166 where the $\dot{\epsilon}_{ij}$ are the strain rates (spatial gradients of the velocity compo-
 167 nents u , v and w) and η is the effective viscosity, which is given by

168

$$\eta = \frac{1}{2} A_g^{-1/n_g} (\dot{\epsilon} + \dot{\epsilon}_0)^{(1-n_g)/n_g} \quad (4)$$

169 As is common practice, the value of the power-law exponent n_g is set to 3
 170 [23]. The minimum strain rate $\dot{\epsilon}_0$ in equation (4) is a mathematical require-
 171 ment that prevents an infinite effective viscosity when $\dot{\epsilon}$, the second invariant
 172 of the strain rate tensor, tends to zero. The flow law rate factor A_g depends
 173 on temperature and (weakly) on ice pressure. Because temperature is taken
 174 as a known quantity here (it is explicit), A_g is also assumed known.

175

176 By assuming that $\partial w/\partial x \ll \partial u/\partial z$ and $\partial w/\partial y \ll \partial v/\partial z$, which is valid
 177 for small aspect ratios [26, 27], and combining equations (1), (2) and (3), the
 178 v and u momentum equations can be written as

$$\begin{aligned}
 & 4 \frac{\partial \eta}{\partial y} \frac{\partial v}{\partial y} + \frac{\partial \eta}{\partial x} \frac{\partial v}{\partial x} + \frac{\partial \eta}{\partial z} \frac{\partial v}{\partial z} + \eta \left(4 \frac{\partial^2 v}{\partial y^2} + \frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial z^2} \right) \\
 & = \rho g \frac{\partial s}{\partial x} - 2 \frac{\partial \eta}{\partial y} \frac{\partial u}{\partial x} - \frac{\partial \eta}{\partial x} \frac{\partial u}{\partial y} - 3\eta \frac{\partial^2 u}{\partial x \partial y}
 \end{aligned} \tag{5}$$

$$\begin{aligned}
 & 4 \frac{\partial \eta}{\partial x} \frac{\partial u}{\partial x} + \frac{\partial \eta}{\partial y} \frac{\partial u}{\partial y} + \frac{\partial \eta}{\partial z} \frac{\partial u}{\partial z} + \eta \left(4 \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} \right) \\
 & = \rho g \frac{\partial s}{\partial x} - 2 \frac{\partial \eta}{\partial x} \frac{\partial v}{\partial y} - \frac{\partial \eta}{\partial y} \frac{\partial v}{\partial x} - 3\eta \frac{\partial^2 v}{\partial x \partial y}
 \end{aligned} \tag{6}$$

179 Equations (5) and (6) are written in "split" form, where all terms on the
 180 left-hand side (LHS) of the equation for v involve v and all terms on the
 181 RHS of the equation for v involve u (and vice versa for the equation for u).
 182 This is the form used by the standard Picard solver in Glimmer-CISM and
 183 the preconditioning operator for our JFNK implementation (the respective
 184 solution procedures are discussed further below).

185

186 2.1. Boundary conditions

187 The free surface boundary conditions for the v and u equations are re-
 188 spectively

$$\left(2\sigma'_{yy}(s) + \sigma'_{xx}(s) \right) \frac{\partial s}{\partial y} + \sigma'_{xy}(s) \frac{\partial s}{\partial x} - \sigma'_{yz}(s) = 0 \tag{7}$$

$$\left(2\sigma'_{xx}(s) + \sigma'_{yy}(s) \right) \frac{\partial s}{\partial x} + \sigma'_{xy}(s) \frac{\partial s}{\partial y} - \sigma'_{xz}(s) = 0 \tag{8}$$

189 Using the constitutive equation, equations (7) and (8) can be written as

$$\left(4\frac{\partial v}{\partial y} + 2\frac{\partial u}{\partial x}\right) \frac{\partial s}{\partial y} + \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right) \frac{\partial s}{\partial x} - \frac{\partial v}{\partial z} = 0 \quad (9)$$

$$\left(4\frac{\partial u}{\partial x} + 2\frac{\partial v}{\partial y}\right) \frac{\partial s}{\partial x} + \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right) \frac{\partial s}{\partial y} - \frac{\partial u}{\partial z} = 0 \quad (10)$$

190 Similarly, the basal boundary conditions for the v and u momentum equa-
191 tions are respectively

$$\tau_{by} = \sigma'_{yz}(b) - \left(2\sigma'_{yy}(b) + \sigma'_{xx}(b)\right) \frac{\partial b}{\partial y} - \sigma'_{xy}(b) \frac{\partial b}{\partial x} \quad (11)$$

$$\tau_{bx} = \sigma'_{xz}(b) - \left(2\sigma'_{xx}(b) + \sigma'_{yy}(b)\right) \frac{\partial b}{\partial x} - \sigma'_{xy}(b) \frac{\partial b}{\partial y} \quad (12)$$

192 where $z = b(x, y)$ defines the lower ice (basal) surface. τ_{by} and τ_{bx} are the
193 components of the basal traction vector. They are formulated here as

194

$$\tau_{by} = -\beta^2 v(b) \quad (13)$$

$$\tau_{bx} = -\beta^2 u(b) \quad (14)$$

195 where $\beta^2 = \beta^2(x, y)$ is a sliding parameter. Its value controls the degree of
196 basal sliding at any location; a very large number ($> 10^6$ Pa s m^{-1}) allows
197 for a quasi no-slip condition, a smaller number ($< 10^2$ Pa s m^{-1}) allows for
198 a moderate amount of basal sliding, and a value of ~ 0 allows for free slip at
199 the ice base, as in the case of a freely floating ice shelf (i.e., water drag at
200 the base is negligible).

201

202 3. Numerical implementation

203 3.1. The nonlinear system of equations

204 A ghost cell approach is used to impose the boundary conditions at the
205 top and at the base. Equations (5) and (6) and the equations related to
206 the boundary conditions are discretized using finite differences and can be
207 written in a compact form as the nonlinear system of n equations at time t ,

$$\mathbf{F}(\mathbf{v}, \mathbf{u}) = \begin{bmatrix} \mathbf{A}_{\mathbf{v}\mathbf{v}}(\mathbf{v}, \mathbf{u}) & \mathbf{A}_{\mathbf{v}\mathbf{u}}(\mathbf{v}, \mathbf{u}) \\ \mathbf{A}_{\mathbf{u}\mathbf{v}}(\mathbf{v}, \mathbf{u}) & \mathbf{A}_{\mathbf{u}\mathbf{u}}(\mathbf{v}, \mathbf{u}) \end{bmatrix} \begin{bmatrix} \mathbf{v} \\ \mathbf{u} \end{bmatrix} - \begin{bmatrix} \mathbf{s}_{\mathbf{v}} \\ \mathbf{s}_{\mathbf{u}} \end{bmatrix} = 0 \quad (15)$$

208 where \mathbf{v} and \mathbf{u} are vectors of size $n/2$, $\mathbf{s}_{\mathbf{v}}$ and $\mathbf{s}_{\mathbf{u}}$ include the purely geometric
 209 terms that do not depend on \mathbf{v} and \mathbf{u} , the matrix $\mathbf{A}_{\mathbf{v}\mathbf{v}}$ ($\mathbf{A}_{\mathbf{u}\mathbf{u}}$) is associated
 210 with the v (u) components of the v (u) equation and the off-diagonal matrix
 211 $\mathbf{A}_{\mathbf{v}\mathbf{u}}$ ($\mathbf{A}_{\mathbf{u}\mathbf{v}}$) is associated with the u (v) components of the v (u) equation.
 212 Notice that the matrices $\mathbf{A}_{\mathbf{v}\mathbf{v}}$, $\mathbf{A}_{\mathbf{u}\mathbf{u}}$, $\mathbf{A}_{\mathbf{v}\mathbf{u}}$ and $\mathbf{A}_{\mathbf{u}\mathbf{v}}$ are functions of \mathbf{v} and \mathbf{u} .
 213 Together, these four $n/2$ block matrices form the $n \times n$ matrix \mathbf{A} . $\mathbf{F}(\mathbf{v}, \mathbf{u})$ is
 214 the residual vector.

215

216 3.2. The Picard solver

217 The Picard solver is based on a splitting of the v and u momentum equa-
 218 tions. To describe this splitting, we first write the system of equations (15)
 219 as

$$\begin{bmatrix} \mathbf{A}_{\mathbf{v}\mathbf{v}}(\mathbf{v}, \mathbf{u}) & 0 \\ \mathbf{0} & \mathbf{A}_{\mathbf{u}\mathbf{u}}(\mathbf{v}, \mathbf{u}) \end{bmatrix} \begin{bmatrix} \mathbf{v} \\ \mathbf{u} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_{\mathbf{v}}(\mathbf{v}, \mathbf{u}) \\ \mathbf{b}_{\mathbf{u}}(\mathbf{v}, \mathbf{u}) \end{bmatrix} \quad (16)$$

220 where $\mathbf{b}_{\mathbf{v}}(\mathbf{v}, \mathbf{u}) = \mathbf{s}_{\mathbf{v}} - \mathbf{A}_{\mathbf{v}\mathbf{u}}\mathbf{u}$ and $\mathbf{b}_{\mathbf{u}}(\mathbf{v}, \mathbf{u}) = \mathbf{s}_{\mathbf{u}} - \mathbf{A}_{\mathbf{u}\mathbf{v}}\mathbf{v}$.

221

222 This can be written as two similar coupled systems of equations of size
 223 $n/2$:

$$\mathbf{A}_{\mathbf{v}\mathbf{v}}(\mathbf{v}, \mathbf{u})\mathbf{v} = \mathbf{b}_{\mathbf{v}}(\mathbf{v}, \mathbf{u}) \quad (17)$$

224 and

$$\mathbf{A}_{\mathbf{u}\mathbf{u}}(\mathbf{v}, \mathbf{u})\mathbf{u} = \mathbf{b}_{\mathbf{u}}(\mathbf{v}, \mathbf{u}) \quad (18)$$

225 The Picard solver in Glimmer-CISM is based on an outer loop, the split-
 226 ting described above and a linear solver. Here is the algorithm of the Picard
 227 solver:

228

- 229 1. Start with an initial iterate $\mathbf{v}^0, \mathbf{u}^0$
- 230 do $k = 1, k_{max}$
- 231 2. Calculate $\eta(\mathbf{v}^{k-1}, \mathbf{u}^{k-1})$
- 232 3. ‘‘Solve’’ $\mathbf{A}_{\mathbf{v}\mathbf{v}}(\mathbf{v}^{k-1}, \mathbf{u}^{k-1})\mathbf{v}^k = \mathbf{b}_{\mathbf{v}}(\mathbf{v}^{k-1}, \mathbf{u}^{k-1})$ using a linear solver

233 4. ‘‘Solve’’ $\mathbf{A}_{uu}(\mathbf{v}^{k-1}, \mathbf{u}^{k-1})\mathbf{u}^k = \mathbf{b}_u(\mathbf{v}^{k-1}, \mathbf{u}^{k-1})$ using a linear solver
234 5. if $\|\mathbf{F}(\mathbf{v}^k, \mathbf{u}^k)\| < \gamma_{nl} \|\mathbf{F}(\mathbf{v}^0, \mathbf{u}^0)\|$ stop
235 enddo

236

237 The initial iterate is the previous time step solution or the zero vector
238 if $t = 0$. γ_{nl} defines the tolerance of the nonlinear solver. In the algorithm
239 above, $\|\cdot\|$ is the L2-norm. $\|\mathbf{F}(\mathbf{v}^0, \mathbf{u}^0)\|$ is the initial nonlinear residual norm.
240 In Glimmer-CISM, the matrix $\mathbf{A}_{vv}(\mathbf{v}^{k-1}, \mathbf{u}^{k-1})$ and the vector $\mathbf{b}_v(\mathbf{v}^{k-1}, \mathbf{u}^{k-1})$
241 are formed in order to solve $\mathbf{A}_{vv}(\mathbf{v}^{k-1}, \mathbf{u}^{k-1})\mathbf{v}^k = \mathbf{b}_v(\mathbf{v}^{k-1}, \mathbf{u}^{k-1})$ for \mathbf{v}^k with
242 a linear solver. The same process is repeated to obtain \mathbf{u}^k . Notice that for
243 step 4, the matrix \mathbf{A}_{uu} and the vector \mathbf{b}_u are formed using \mathbf{v}^{k-1} not \mathbf{v}^k . When
244 close to a converged solution, the ‘‘unstable manifold correction’’ scheme of
245 [30] may also be applied after step 4. The standard linear solver, which is part
246 of the SLAP package [36], is the Generalized Minimum RESidual (GMRES,
247 [37]) method preconditioned by an Incomplete LU (ILU) factorization [38].
248 The criterion for convergence in step 3 is (an analogous criterion is applied
249 to step 4) :

$$\frac{\|\mathbf{P}_{\text{ILU}}^{-1}(\mathbf{b}_v - \mathbf{A}_{vv}\mathbf{v}^k)\|}{\|\mathbf{P}_{\text{ILU}}^{-1}\mathbf{b}_v\|} < \gamma_l \quad (19)$$

250 where $\mathbf{P}_{\text{ILU}}^{-1}$ is the preconditioning operator for the Picard solver.

251 3.3. The JFNK solver

252 Unlike the Picard solver in Glimmer-CISM, the JFNK method does not
253 split the v and u equations. However, there is a splitting in the preconditioning
254 step because we use the Picard linear solver for this operation. We
255 introduce the vector $\mathbf{x} = [\mathbf{v} \ \mathbf{u}]^T$, which is formed by stacking the v components
256 of velocity followed by the u components. We then have the same
257 nonlinear system of equations (15) to solve as discussed previously but written as
258

$$\mathbf{F}(\mathbf{x}) = \mathbf{F}(\mathbf{v}, \mathbf{u}) = \begin{bmatrix} \mathbf{F}_v(\mathbf{v}, \mathbf{u}) \\ \mathbf{F}_u(\mathbf{v}, \mathbf{u}) \end{bmatrix} = 0 \quad (20)$$

259 The Newton method is based on a multivariate Taylor expansion around
260 a previous iterate (\mathbf{x}^{k-1}):

$$\mathbf{F}(\mathbf{x}^{k-1} + \delta\mathbf{x}^k) \approx \mathbf{F}(\mathbf{x}^{k-1}) + \mathbf{F}'(\mathbf{x}^{k-1})\delta\mathbf{x}^k \quad (21)$$

261 where the higher order terms are neglected in the Taylor expansion.

262

263 Setting $\mathbf{F}(\mathbf{x}^{k-1} + \delta\mathbf{x}^k) = 0$, the correction $\delta\mathbf{x}^k = \mathbf{x}^k - \mathbf{x}^{k-1}$ can be obtained
 264 by solving the linear system of n equations, here using a Krylov method:

$$\mathbf{J}(\mathbf{x}^{k-1})\delta\mathbf{x}^k = -\mathbf{F}(\mathbf{x}^{k-1}) \quad (22)$$

265 where the system matrix $\mathbf{J} \equiv \mathbf{F}'$ is the Jacobian, an $n \times n$ matrix with ele-
 266 ments given by $J_{ij} = \partial F_i(\mathbf{x}^{k-1})/\partial(x_j^{k-1})$ (with $i = 1, n$ and $j = 1, n$).

267

268 The Newton-Krylov algorithm is:

269

```

270 1. Start with an initial iterate  $\mathbf{x}^0$ 
271 do  $k = 1, k_{max}$ 
272   2. Calculate  $\eta(\mathbf{x}^{k-1})$ 
273   3. ‘‘Solve’’  $\mathbf{J}(\mathbf{x}^{k-1})\delta\mathbf{x}^k = -\mathbf{F}(\mathbf{x}^{k-1})$  using a Krylov method
274   4.  $\mathbf{x}^k = \mathbf{x}^{k-1} + \delta\mathbf{x}^k$ 
275   5. if  $\|\mathbf{F}(\mathbf{x}^k)\| < \gamma_{nl} \|\mathbf{F}(\mathbf{x}^0)\|$  stop
276 enddo

```

277

278 As with Picard, the initial iterate is the previous time step solution or
 279 the zero vector if $t = 0$. In step 3, the convergence criterion for the linear
 280 solver is $\|\mathbf{J}(\mathbf{x}^{k-1})\delta\mathbf{x}^k + \mathbf{F}(\mathbf{x}^{k-1})\| < \gamma_l(k) \|\mathbf{F}(\mathbf{x}^{k-1})\|$. The parameter $\gamma_l(k)$,
 281 which is a constant smaller than unity, defines the tolerance of the linear
 282 solver. Solving $\mathbf{J}(\mathbf{x}^{k-1})\delta\mathbf{x}^k = -\mathbf{F}(\mathbf{x}^{k-1})$ to a very high accuracy ($\gamma_l(k) \rightarrow 0$)
 283 might overall increase the total CPU time in finding the nonlinear approxi-
 284 mate solution and might even decrease the robustness of the solver. When
 285 the approximate solution to the linear system of equations is not ‘‘accurate’’,
 286 the approach is referred to as an inexact Newton method [39]. This is the
 287 approach adopted in this work, as discussed further below.

288

289 Our implementation of JFNK relies on code re-use from the existing
 290 Picard solver within the Glimmer-CISM model. First, the residual vector
 291 $\mathbf{F}(\mathbf{x}^{k-1})$ can be obtained as

$$\mathbf{F}(\mathbf{x}^{k-1}) = \begin{bmatrix} \mathbf{A}_{\mathbf{v}\mathbf{v}}(\mathbf{x}^{k-1})\mathbf{v}^{k-1} - \mathbf{b}_{\mathbf{v}}(\mathbf{x}^{k-1}) \\ \mathbf{A}_{\mathbf{u}\mathbf{u}}(\mathbf{x}^{k-1})\mathbf{u}^{k-1} - \mathbf{b}_{\mathbf{u}}(\mathbf{x}^{k-1}) \end{bmatrix} \quad (23)$$

292 To calculate the residual vector \mathbf{F} , we just had to code a simple MATVEC

293 subroutine as we reuse the part of the Glimmer-CISM code that calculates
 294 the matrices $\mathbf{A}_{\mathbf{v}\mathbf{v}}$ and $\mathbf{A}_{\mathbf{u}\mathbf{u}}$ and the vectors $\mathbf{b}_{\mathbf{v}}$ and $\mathbf{b}_{\mathbf{u}}$. As opposed to the
 295 Picard solver implementation which requires only one matrix to be stored
 296 (because of the splitting), both matrices $\mathbf{A}_{\mathbf{v}\mathbf{v}}(\mathbf{x}^{k-1})$ and $\mathbf{A}_{\mathbf{u}\mathbf{u}}(\mathbf{x}^{k-1})$ are stored
 297 for JFNK because they are used for the preconditioning step.

298

299 Obtaining the Jacobian matrix is a difficult development task for compli-
 300 cated problems such as the one considered here. Moreover, forming this sys-
 301 tem matrix could be computationally expensive. For these reasons, we adopt
 302 a Jacobian-free approach, which is possible when using a Krylov method as
 303 the linear solver. Indeed, Krylov methods approximate the linear solution in
 304 a subspace of the form $(\mathbf{r}_0, \mathbf{J}\mathbf{r}_0, \mathbf{J}^2\mathbf{r}_0\dots)$ where \mathbf{r}_0 is the initial linear residual
 305 given by $\mathbf{J}(\mathbf{x}^{k-1})\delta\mathbf{x}_0^k + \mathbf{F}(\mathbf{x}^{k-1})$ [39], with an initial guess $\delta\mathbf{x}_0^k$ usually taken
 306 to be zero. Because Krylov methods require only the product of the system
 307 matrix (the Jacobian) and a vector, the Jacobian matrix does not need to be
 308 formed and stored explicitly. Rather, only its action on a vector is required.
 309 This property is fundamental for the implementation of a Jacobian-free ap-
 310 proach, which relies on the fact that the product of \mathbf{J} times a vector \mathbf{w} can
 311 be approximated by a first-order Taylor series expansion

$$\mathbf{J}(\mathbf{x}^{k-1})\mathbf{w} \sim \frac{\mathbf{F}(\mathbf{x}^{k-1} + \epsilon\mathbf{w}) - \mathbf{F}(\mathbf{x}^{k-1})}{\epsilon}, \quad (24)$$

312 where ϵ is a small number (10^{-7} in our implementation).

313

314 The approximation of \mathbf{J} times a vector can be calculated using the ma-
 315 chinery previously described in equation (23).

316

317 Krylov methods for solving linear systems of stiff equations are likely to
 318 converge very slowly and to exhibit robustness issues unless preconditioning
 319 is applied [40]. By using preconditioning, one solves an equivalent system
 320 of equations that has the same solution as the original system but which
 321 is numerically easier to solve. In this work we use the Flexible GMRES
 322 (FGMRES) approach which relies on right preconditioning [40]. In this case,
 323 equation (22) becomes

$$\mathbf{J}(\mathbf{x}^{k-1})\mathbf{P}_{\mathbf{p}}^{-1}\delta\mathbf{z} = -\mathbf{F}(\mathbf{x}^{k-1}) \quad (25)$$

324 where $\delta\mathbf{z} = \mathbf{P}_{\mathbf{p}}\delta\mathbf{x}^k$ and $\mathbf{P}_{\mathbf{p}}^{-1}$ is referred to as the preconditioning operator.

325 The subscript \mathbf{p} indicates that the Picard solver is used for the JFNK pre-
 326 conditioning operator (as discussed further below).

327

328 Note that $\mathbf{P}_{\mathbf{p}}^{-1}$ is an operator that should not be necessarily considered
 329 as the inverse of a matrix. We are looking for an operator that *approximates*
 330 the inverse of the system matrix (\mathbf{J}^{-1}). As before, we are looking for
 331 a Jacobian-free approach.

332

333 The matrix \mathbf{J} can be divided into two matrices [35]

$$\mathbf{J}(\mathbf{x}^{k-1}) = \mathbf{A}(\mathbf{x}^{k-1}) + \mathbf{G}(\mathbf{x}^{k-1}) \quad (26)$$

334 where $\mathbf{A}(\mathbf{x}^{k-1})$ is the matrix described in equation (15, linearized with \mathbf{x}^{k-1})
 335 and $\mathbf{G}(\mathbf{x}^{k-1})$ is a matrix with entries made up of sums of terms like
 336 $x_j^{k-1} \partial a_{ij}(\mathbf{x}^{k-1}) / \partial (x_j^{k-1})$ (the a_{ij} being the elements of the matrix \mathbf{A}). The
 337 matrix \mathbf{G} reflects the nonlinear nature of the ice sheet rheology, i.e., the fact
 338 that the viscosity depends on the velocity field as opposed to a Newtonian
 339 fluid. Using the block matrices introduced in equation (15), we can write
 340 $\mathbf{J}(\mathbf{x}^{k-1})$ as

$$\mathbf{J}(\mathbf{x}^{k-1}) = \begin{bmatrix} \mathbf{A}_{\mathbf{vv}}(\mathbf{x}^{k-1}) & 0 \\ 0 & \mathbf{A}_{\mathbf{uu}}(\mathbf{x}^{k-1}) \end{bmatrix} + \begin{bmatrix} 0 & \mathbf{A}_{\mathbf{vu}}(\mathbf{x}^{k-1}) \\ \mathbf{A}_{\mathbf{uv}}(\mathbf{x}^{k-1}) & 0 \end{bmatrix} + \mathbf{G}(\mathbf{x}^{k-1}) \quad (27)$$

341 While the off-diagonal matrices $\mathbf{A}_{\mathbf{vu}}(\mathbf{x}^{k-1})$ and $\mathbf{A}_{\mathbf{uv}}(\mathbf{x}^{k-1})$ are never formed
 342 within the existing Picard solver and getting $\mathbf{G}(\mathbf{x}^{k-1})$ is very complicated
 343 (the reason why the system matrix $\mathbf{J}(\mathbf{x}^{k-1})$ is so difficult to obtain), we do
 344 have access to the matrices $\mathbf{A}_{\mathbf{vv}}(\mathbf{x}^{k-1})$ and $\mathbf{A}_{\mathbf{uu}}(\mathbf{x}^{k-1})$. It is therefore possible
 345 to use these two matrices for the preconditioning operator. Given a vector
 346 \mathbf{q} , the preconditioning operator applied to it leads to the vector $\mathbf{r} = \mathbf{P}_{\mathbf{p}}^{-1} \mathbf{q}$
 347 where \mathbf{q} and \mathbf{r} are vectors formed during the Krylov iteration process. In
 348 more details, based on the splitting approach of the Picard solver and the
 349 matrices $\mathbf{A}_{\mathbf{vv}}(\mathbf{x}^{k-1})$ and $\mathbf{A}_{\mathbf{uu}}(\mathbf{x}^{k-1})$, the preconditioning operator leads to

$$\mathbf{r}_{\mathbf{v}} = \tilde{\mathbf{A}}_{\mathbf{vv}}^{-1} \mathbf{q}_{\mathbf{v}} \quad (28)$$

$$\mathbf{r}_{\mathbf{u}} = \tilde{\mathbf{A}}_{\mathbf{uu}}^{-1} \mathbf{q}_{\mathbf{u}} \quad (29)$$

350 where $\mathbf{q} = [\mathbf{q}_v \ \mathbf{q}_u]^T$ and $\mathbf{r} = [\mathbf{r}_v \ \mathbf{r}_u]^T$, and the tilde indicates that we solve
 351 equations (28) and (29) to a very loose tolerance ($\gamma_l = 10^{-3}$, see equation (19)
 352 for details). In other words, \mathbf{r}_v is the approximate solution of $\mathbf{A}_{vv}\mathbf{t}_v = \mathbf{q}_v$
 353 where \mathbf{t}_v would be the exact solution.

354

355 3.4. Robustness of the solvers

356 For the Picard solver, γ_l in equation (19) is set to 10^{-12} . This is a very
 357 small value of γ_l . Tests have shown that this high tolerance increases the
 358 robustness of the Picard solver [28]. Note that the same criterion is used for
 359 the u equation in step 4.

360

361 We have included two small modifications to our JFNK solver to improve
 362 its robustness. As was observed by [41] and [35], a small value of the forcing
 363 term γ_l (see equation (30)) in early Newton iterations increases the CPU
 364 time and can even prevent convergence of the approximate solution. We
 365 have substantially improved the robustness of our JFNK solver by using an
 366 inexact Newton method. Following [42], the value of the parameter $\gamma_l(k)$ in
 367 our inexact Newton approach depends on previous values of the L2-norm. It
 368 is given by

$$\gamma_l(k) = \alpha \left(\frac{\|\mathbf{F}(\mathbf{x}^{k-1})\|}{\|\mathbf{F}(\mathbf{x}^{k-2})\|} \right)^m \quad (30)$$

369 where the parameters α and m are 1 and 2, respectively. We have not done
 370 a thorough investigation to optimize these parameters. For the first Newton
 371 iteration, $\gamma_l(k) = 0.9$ and it is limited to $0.01 \leq \gamma_l(k) \leq 0.9$ for subsequent it-
 372 erations. As discussed later, this progressive tolerance (equation (30)) might
 373 only be needed for a few time steps at the beginning of the run when the
 374 initial iterate is far from the solution. Later in the simulation, a more aggres-
 375 sive tolerance (e.g., $\gamma_l(k) = 0.01$) could be used to improve the computational
 376 efficiency without affecting the robustness.

377

378 The second modification is related to the basal boundary condition. Strong
 379 friction at the base ($\beta^2 \gg 1$) leads to large off-diagonal elements in the \mathbf{A}_{vv}
 380 and \mathbf{A}_{uu} matrices. A rescaling of the elements in \mathbf{A}_{vv} and \mathbf{A}_{uu} associated
 381 with the basal boundary conditions has proven to significantly improve the
 382 robustness of the JFNK solver. Note that the same scaling is used here for

test case	domain	problem size	β^2 (Pa s m ⁻¹)
dome-2km	30x30x10	6640	10^{10}
dome-1km	60x60x10	27760	10^{10}
dome-0.5km	120x120x10	111520	10^{10}
dome-0.25km	240x240x10	448960	10^{10}
ISMIP-HOM-C	91x91x11	182116	variable and periodic
shelf	51x51x11	48576	10^{-5}
GIS-20km	76x141x11	94644	10^{10}
GIS-10km	151x281x11	371602	10^{10}
GIS-5km	301x561x11	1483196	10^{10}

Table 1: The different test cases

383 the Picard solver but that it does not affect its robustness and performances.
384

385 4. The test cases

386 We have used a suite of test cases to assess the robustness and compu-
387 tational efficiency of the JFNK method and to compare these to the Picard
388 solver. A wide range of problem sizes, configurations, and boundary condi-
389 tions are covered. Table 1 gives relevant details about the different test cases.
390

391 The name of each test case is given in the first column of Table 1. The
392 horizontal (East-West and North-South) dimensions and the number of ver-
393 tical levels are given in the second column and the size of the problem is given
394 in the third column. The size of the problem is defined as the number of grid
395 cells where ice is present (at the beginning of the simulation) times two (for
396 the two components of velocity). The last column gives some information
397 about the sliding coefficient β^2 .
398

399 The first four test cases correspond to the same problem: a parabolic
400 dome of ice with a circular, 60 km diameter base. The horizontal spatial
401 resolutions studied are 2 km, 1 km, 0.5 km and 0.25 km, and there are 10
402 vertical levels. For this set of experiments, a quasi no-slip basal condition
403 is imposed by setting $\beta^2 = 10^{10}$ Pa s m⁻¹. A zero-flux boundary condition
404 is applied at the dome margins. We refer to this set of experiments as the

ρ	ice density	910 kg m^{-3}
g	gravitational acceleration	9.81 m s^{-2}
n_g	exponent in Glen's law	3
A_g	flow rate parameter	$10^{-16} \text{ Pa}^{-n_g} \text{ yr}^{-1}$

Table 2: Physical constants used in the simulations

405 “dome” test cases. The next test case (“shelf”) involves a flat ice shelf of
406 uniform thickness at flotation that is confined on three sides by solid walls
407 but open to the ocean on the fourth side. Along the confined walls a zero-flux
408 boundary condition is applied and along the open front the stress is balanced
409 by the hydrostatic pressure due to a column of ocean water. The drag on
410 the shelf bottom is negligible as β^2 is set to 10^{-5} . In the ISMIP-HOM-C test
411 case, the β^2 value is doubly periodic as described in [29]. The last set of test
412 cases represent the Greenland ice sheet (GIS) at different spatial resolutions
413 (20 km, 10 km and 5 km), based on the 5 km digital elevation model of [43].
414 A quasi-no slip boundary condition is applied at the bed. As with the dome
415 test cases, a zero-flux boundary condition is applied at the lateral margins.
416 We refer to these respectively as GIS-20km, GIS-10km and GIS-5km. In all
417 test cases, the ice is taken as isothermal with a constant and uniform rate
418 factor of $10^{-16} \text{ Pa}^{-n_g} \text{ yr}^{-1}$.

419

420 Table 2 gives the value of the physical constants used by the model for
421 the simulations presented here.

422

423 We treat the evolution of the ice thickness using the “incremental remap-
424 ping” scheme of [44], a higher-order advection scheme that has been applied
425 successfully to the thickness evolution of sea ice [45].

426

427 5. Results

428 All simulations were performed on a desktop computer (Intel(R) dual-
429 core(TM) E8400 3.00 GHz CPU, cache of 6144 KB with a RAM of 4 Gb).
430 The fortran compiler is gfortran 4.1.2, 64 bits and the O3 optimization is
431 used. We use revision 2002 of a Glimmer-CISM developmental branch (called
432 the parallel branch) that will be released to the public in the near future.

test case	ite JFNK	ite Picard	CPU JFNK	CPU Picard	gain
dome-2km	11	51	1.39	3.13	2.25
ISHOM-HOM-C	15	58	170.97	522.81	3.06
shelf	8	60	37.09	101.42	2.73
GIS-20km	10	51	17.89	45.01	2.51
GIS-10km	15	55	136.36	251.42	1.84

Table 3: Number of outer iterations and CPU time required to calculate a diagnostic velocity field when using either the Picard solver or the JFNK solver and computational gain of JFNK over Picard depending on the test case.

433

434 *5.1. Robustness*

435 In the first series of experiments, we assess the robustness of the JFNK
436 and Picard solvers in calculating a diagnostic velocity field. The initial it-
437 erate is the zero vector, which provides a good test to evaluate the global
438 convergence¹ properties (and therefore robustness) of a nonlinear solver as
439 this initial iterate is far from the solution. Changing the size of the time
440 step for a diagnostic solution does not affect the robustness results presented
441 here because the inertial term is neglected in the momentum equations (i.e.
442 a steady-state solution given a temperature and thickness fields). For all the
443 following robustness experiments, the nonlinear tolerance (γ_{nl}) is set to 10^{-8} .

444

445 Figure 1 shows the evolution of the L2-norm for the Picard solver (dashed
446 curves) and the JFNK solver (solid curves) with the number of outer itera-
447 tions. Table 3 summarizes the results for the number of outer iterations and
448 CPU time required to get the approximate solution. The current focus is ro-
449 bustness so we assess if the solvers are able (or not) to calculate the required
450 approximate solution. We also give the number of iterations and CPU time
451 required for the simulations, but given that the first step is a very particular
452 case, the iteration count and computational gain of JFNK over Picard might
453 not be representative.

454

¹an iterative method is said to be globally convergent if the approximate solution converges to the true solution given some arbitrary initial iterate

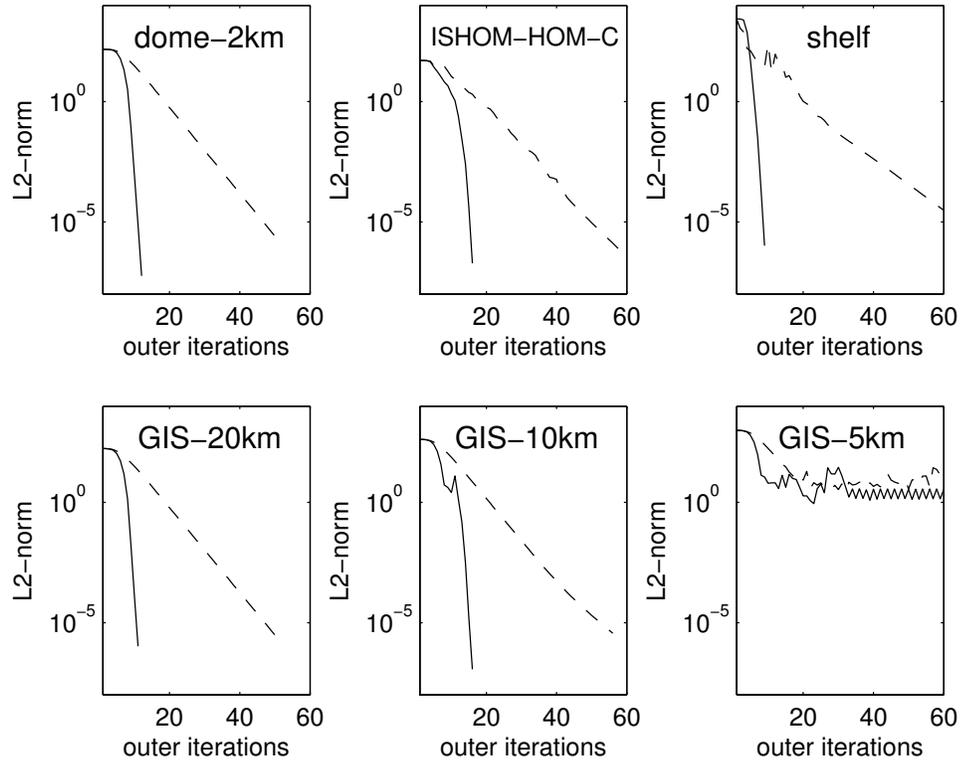


Figure 1: L2-norm as a function of the number of outer loop iterations for the Picard solver (dashed curves) and the JFNK solver (solid curves). The test cases are: a) dome-2km, b) ISHOM-HOM-C c) shelf d) GIS-20 km, e) GIS-10 km and f) GIS-5 km.

455 Both solvers converge to the solution for the dome-2km, ISHOM-HOM-C,
 456 shelf, GIS-20km and GIS-10km test cases. However, Picard and JFNK fail
 457 to converge at 5 km resolution for the Greenland ice sheet test case. For both
 458 solvers, the residual initially decreases by almost three orders of magnitude
 459 but then levels off. A plot of the residual on the grid shows that this conver-
 460 gence issue only affects a few grid cells (not shown). We discuss several
 461 potential strategies for addressing this problem in the conclusions.

462

463 The JFNK solver requires between 3.7 to 7.5 times fewer outer iterations
 464 than the Picard solver to get the solution. Nevertheless, because a Newton

465 iteration involves more computations, the computational gain is not as high
466 as the iteration count ratio. Here, we find that the JFNK solver is between
467 1.84-3.06 times faster than the Picard solver depending on the particular test
468 case.

469

470 5.2. Computational efficiency

471 Because the initial iterate is far from the solution, both solvers have a
472 harder time solving the nonlinear system of equations at the beginning of
473 the runs (for a few time steps). After this transition phase, the behavior of
474 each solver is fairly constant (i.e., the CPU time and number of outer iter-
475 ations required per time step do not vary significantly). The main goal in
476 developing a JFNK solver is to significantly reduce the computational time
477 for long-term simulations. In this section, we assess the computational gain
478 of JFNK over the Picard solver in the context of long-term simulations by
479 looking at the performances of both solvers after the initial transition phase.
480 We are also interested in understanding how this computational gain evolves
481 as the problem size increases.

482

483 To this end, we perform two sets of experiments. In the first one, we
484 compare the computational efficiency of JFNK to the one of Picard for the
485 dome test cases with increasing horizontal spatial resolution. In the second
486 set of experiments, we investigate the performance of JFNK and Picard in
487 simulating the Greenland ice sheet (under the conditions given above) at 20
488 km and 10 km resolutions. For all these experiments, $\gamma_{nl} = 10^{-6}$ and the
489 time step is 1 year. The simulations are run for 20 years. The mean CPU
490 time per time step required by Picard is compared to the mean CPU time
491 needed by JFNK. The mean values are calculated based on time steps 11 to
492 20 (i.e. sufficiently far from the initial iterate).

493

494 As an aside, Figure 2 shows a typical velocity field solution. It is the sur-
495 face ice speed calculated with the JFNK solver for the GIS-10km test case
496 at $t = 10$ years. In accordance with observations, the fastest velocities are
497 found close to ice margins.

498

499 Figure 3 shows the relative computational gain of JFNK versus Picard
500 as a function of the problem size for the dome test cases (solid curve with *
501 data points) and the Greenland ice sheet experiments (the + data points).

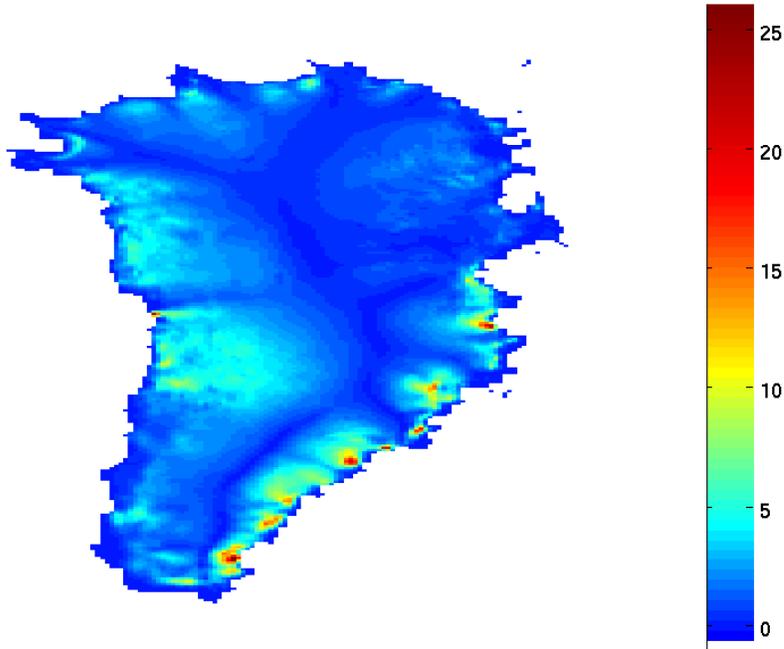


Figure 2: Surface ice speed calculated with the JFNK solver for the GIS-10km test case at $t = 10$ years. Values are in m/year.

502 The computational gain is calculated as the mean CPU time per time step
 503 required by Picard divided by the mean CPU time per time step for JFNK.
 504 The JFNK solver is clearly more efficient than Picard for all the spatial
 505 resolutions tested and for both sets of experiments. Furthermore, the com-
 506 putational gain increases as the grid is refined. For the dome experiments,
 507 JFNK is 2.6 times faster than Picard for a 2 km resolution. This gain in-
 508 creases to 3.62 at 0.25 km resolution. Although the computational gains
 509 are lower, JFNK is still significantly faster than Picard for the Greenland
 510 experiments: the gain is 2.14 at 20 km and 2.44 at 10 km resolution. The
 511 computational gain of JFNK over the Picard solver also increases when a
 512 more accurate solution (a smaller γ_m) is required (results not shown). Note
 513 that these numbers (the computational gain) are slightly biased in favor of

514 the Picard solver because the final L2-norm of the JFNK solver is often sig-
 515 nificantly lower than that from the Picard solver; from one iteration to the
 516 next, the L2-norm drops by a larger amount for a single Newton iteration
 517 than for a single Picard iteration (see Figure 5 for an example of this).
 518

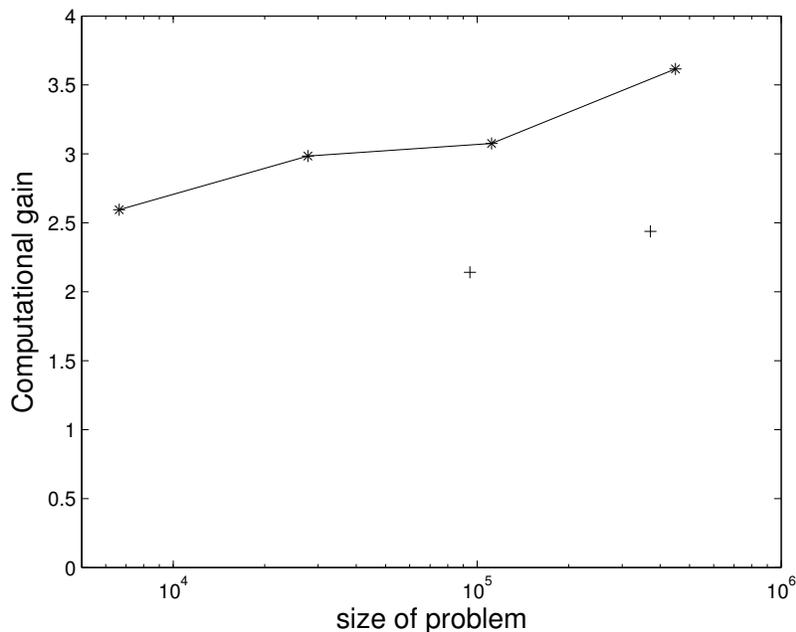


Figure 3: Mean CPU time needed by Picard divided by the mean CPU time for JFNK as a function of the problem size for the dome test cases (*) and the Greenland ice sheet simulations (+).

519 Figure 4 gives additional information on the behavior of JFNK as the grid
 520 is refined. First, Figure 4a shows on a log-log plot how the CPU time evolves
 521 as the resolution is increased. The relationship between the CPU time and
 522 the problem size is linear (on a log-log plot) but the slope is slightly larger
 523 than 1 (for both sets of experiments). The “extra work” is performed by
 524 the inner GMRES (Figure 4c) as the number of FGMRES iterations is fairly
 525 constant when changing the spatial resolution (Figure 4b). At 2 km resolu-
 526 tion with the “dome” test case, the percentage of CPU time associated with
 527 the preconditioning operator is slightly larger than 25% while this percentage
 528 increases to 45% with a 0.25 km grid. The use of a more efficient precondi-

529 tioner such as an algebraic multigrid will likely result in better scaling (slope
 530 close to 1).
 531

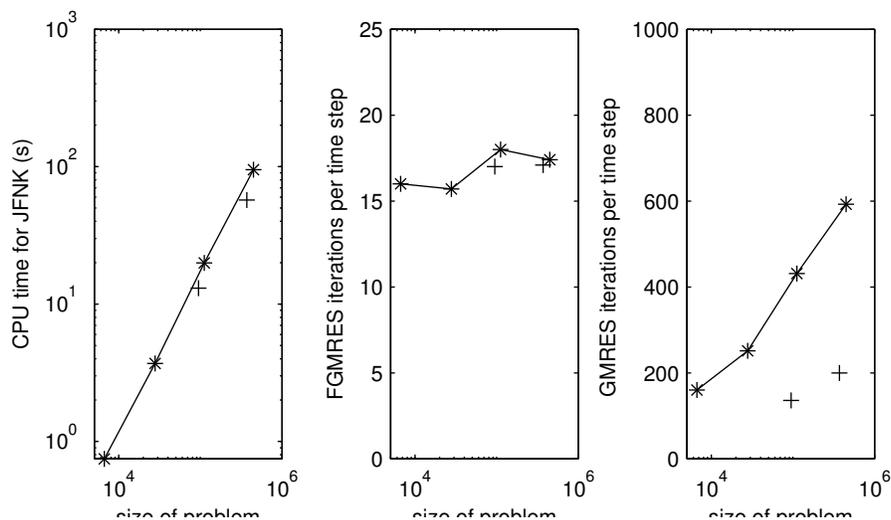


Figure 4: a) Average CPU time per time step required by JFNK as a function of the problem size. b) Average number of FGMRES iterations per time step for the JFNK solver as a function of the problem size. c) Average number of GMRES (the preconditioner) iterations per time step for the JFNK solver as a function of the problem size. The data points for the “dome” test cases are the * while the + correspond to the Greenland ice sheet simulations.

532 Figure 5 shows a typical evolution of the L2-norm of the nonlinear system
 533 of equations when using either JFNK or the Picard solver for the GIS-10km
 534 test case at $t = 10$ years. The L2-norm as a function of the number of outer
 535 iterations for the Picard solver is shown with the dashed curve. The solid
 536 curve with the * data points shows the L2-norm with the number of New-
 537 ton iterations for JFNK when using the progressive linear tolerance given by
 538 equation (30) (the default approach). Because this progressive linear toler-
 539 ance is only needed for the first few time steps to increase robustness, more
 540 aggressive tolerance values can be used for later times. We have done this here
 541 by setting $\gamma_l(k) = 0.01$ at $t = 10$ years, for the solid curve with the + data
 542 points on Figure 5. With this tighter tolerance, JFNK only needs 3 Newton
 543 iterations to reach the convergence criterion while 5 Newton iterations are
 544 required with the progressive tolerance. In terms of computational efficiency,
 545 JFNK is 2.45 faster than Picard with the progressive tolerance and 2.95 faster

546 with a fixed $\gamma_l(k)$ of 0.01. As expected from theory, the convergence rate of
 547 Picard is linear. For JFNK, the convergence rate is not quadratic because an
 548 inexact Newton approach is used. Asymptotic quadratic convergence could
 549 be achieved by using very small values of $\gamma_l(k)$ [39].

550

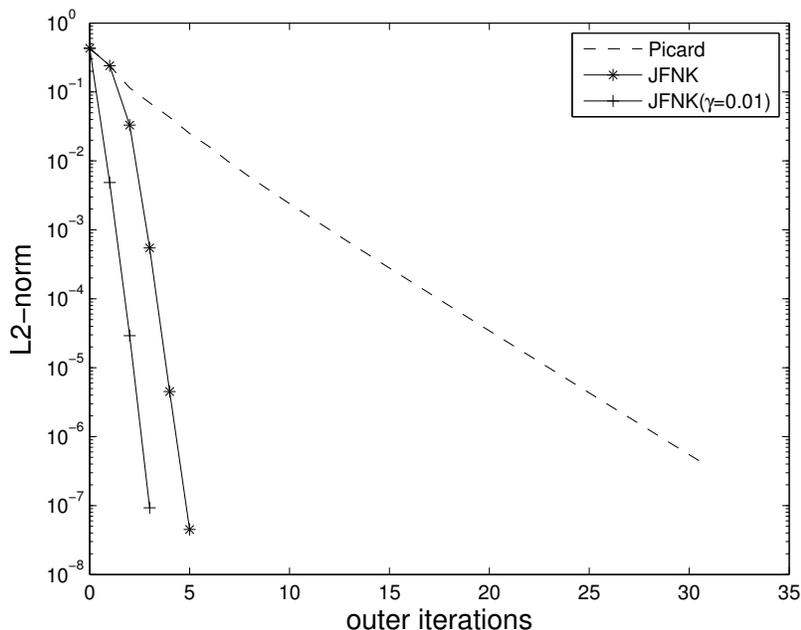


Figure 5: L2-norm as a function of the number of outer iterations for the Picard solver, JFNK with $\gamma_l(k)$ given by equation (30) and JFNK with $\gamma_l(k) = 0.01$. This is for the GIS-10km test case at $t = 10$ years.

551 The use of the Picard solver (GMRES+ILU) as a preconditioner has lead
 552 to a quick implementation of the JFNK solver. One might argue however
 553 that a more computationally efficient approach would be to simply use ILU
 554 for the preconditioning step and therefore put aside the inner GMRES. Re-
 555 sults indicate that at low resolution, JFNK with ILU as the preconditioner
 556 (referred here as JFNK-ILU) is as efficient as our standard JFNK (with
 557 GMRES+ILU for the preconditioning step). However, as the resolution is
 558 increased, JFNK-ILU is less and less efficient (results not shown). This is
 559 caused by a significant increase in the number of FGMRES iterations when

560 compared to our standard JFNK approach. This is detrimental on the CPU
561 time as function evaluations in the approximation of the Jacobian times a
562 vector (equation (24)) are costly. As it was observed by others, ILU does
563 not perform well when refining the grid as it leads to a lot more of Krylov
564 iterations (e.g., [46]).

565

566 6. Conclusion

567 We have implemented a Jacobian-Free Newton-Krylov (JFNK) method
568 to solve the first-order ice sheet momentum equations. This new solver has
569 been implemented into the framework of the Community Ice Sheet Model
570 (Glimmer-CISM), the land ice component of the Community Earth System
571 Model (CESM). It was developed by re-using many parts of the existing
572 Picard solver in Glimmer-CISM. Specifically, we have developed a physics
573 based preconditioner with the existing Glimmer-CISM Picard solver. The
574 re-use of previously tested code has led to a quick implementation.

575

576 Two minor modifications considerably improved the robustness of our
577 JFNK implementation. First, we apply a loose linear tolerance during the
578 early stage of the Newton process. As observed by others (e.g., [41], [35]),
579 the use of a tight tolerance in the early Newton iterations can be detrimental
580 in terms of CPU time and can even prevent the method from converging. A
581 loose tolerance of $\gamma_l(k) = 0.9$ is applied for the first Newton iteration and
582 subsequent values are calculated according to the evolution of the L2-norm
583 (based on [42]). Second, the imposition of the boundary condition at the
584 base can lead to very large matrix coefficients. To improve the robustness,
585 a rescaling of the rows of the matrix associated with this basal boundary
586 condition is employed.

587

588 With these modifications, JFNK converges for a range of different test
589 cases even when starting with an initial iterate that is far from the solution.
590 However, our tests also indicate that JFNK sometimes does not converge
591 for high resolution simulations. We note that the existing Picard solver also
592 exhibits a lack of robustness for these same fine grid test cases. Additional
593 work needs to be done to eliminate this problem. A parameter continuation
594 method [39] has not been able to resolve this issue. This was done by slowly
595 adjusting the value of the exponent in Glen’s law (equation (3)) during the

596 Newton iteration. We are currently investigating the use of a combination
597 Picard (in the early stage) and Newton (at the end of the process) solver to
598 improve the overall solver robustness [32]. We are also investigating homo-
599 topy and pseudo-transient continuation [39] methods as possible globalization
600 approaches.

601

602 The computational efficiency of JFNK (and its comparison to the one of
603 Picard) was studied for two different sets of experiments with variable hori-
604 zontal spatial resolution. These tests indicate that JFNK is between 2.14 to
605 3.62 times faster than the Picard solver. The computational gain of JFNK
606 over Picard increases as the grid is refined. Moreover, the computational
607 gain increases when a tighter nonlinear tolerance is required. In other words,
608 the JFNK solver is increasingly more efficient compared to Picard when a
609 more accurate solution is desired. The JFNK method is therefore a signif-
610 icant improvement in terms of computational efficiency when compared to
611 the standard solver of Glimmer-CISM (the Picard solver).

612

613 Because the percentage of CPU time associated with the preconditioning
614 step increases with the spatial resolution of the problem, a more efficient pre-
615 conditioner, such as algebraic multigrid, could further improve the results.
616 We are currently implementing the Trilinos solver package [47], after which
617 we will replace our “homemade” JFNK solver with the Trilinos NOX solver,
618 using multi-level (ML) for the preconditioning step. We are also developing
619 a multi-pass Picard preconditioner (which involves the use of the off-diagonal
620 block matrices) to reduce the CPU time associated with the preconditioning
621 operator. Current work also includes a complete parallelization of the model.

622

623 **Acknowledgments**

624 We would like to thank Carl Gladish for his help in setting up and compil-
625 ing Glimmer-CISM and William H. Lipscomb for helpful discussions. Jean-
626 François Lemieux is grateful to FQRNT and NSERC for Postdoctoral fel-
627 lowships. This work has been funded by the Department of Energy Office
628 of Advanced Scientific Computing project, A Scalable, Efficient, and Accu-
629 rate Community Ice Sheet Model, within the ISICLES initiative. Sandia is a
630 multiprogram laboratory operated by Sandia Corporation, a Lockheed Mar-
631 tin Company, for the United States Department of Energy under contract

632 DE-AC04-94AL85000.

633

634 **References**

- 635 [1] I. Joughin, A. W, F. M, Large fluctuations in speed on Green-
636 land’s Jakobshavn Isbrae glacier, *Nature* 432 (2004) 608–610,
637 doi:10.1038/nature03130.
- 638 [2] I. M. Howat, I. Joughin, T. S, S. Gogineni, Rapid retreat and accelera-
639 tion of Helheim glacier, east Greenland, *Geophys. Res. Lett.* 32 (2005)
640 L22502, doi:10.1029/2005GL024737.
- 641 [3] E. Rignot, P. Kanagaratnam, Changes in the velocity struc-
642 ture of the Greenland ice sheet, *Science* 311 (2006) 986–990,
643 doi:10.1126/science.1121381.
- 644 [4] I. M. Howat, I. Joughin, T. A. Scambos, Rapid changes in ice discharge
645 from Greenland outlet glaciers, *Science* 315 (5818) (2007) 1559–1561,
646 dOI: 10.1126/science.1138478.
- 647 [5] A. Luckman, T. Murray, R. de Lange, E. Hanna, Rapid and synchronous
648 ice-dynamic changes in East Greenland, *Geophys. Res. Lett.* 33 (2006)
649 L03503, doi:10.1029/2005GL025428.
- 650 [6] T. Murray, K. Scharrer, T. D. James, S. R. Dye, E. Hanna, A. D. Booth,
651 N. Selmes, A. Luckman, A. L. C. Hugues, S. Cook, P. Huybrechts, Ocean
652 regulation hypothesis for glacier dynamics in southeast Greenland and
653 implications for ice sheet mass changes, *J. Geophys. Res.* 115 (2010)
654 F03026, doi:10.1029/2009JF001522.
- 655 [7] H. De Angelis, P. Skvarca, Glacier surge after ice shelf collapse, *Science*
656 299 (5612) (2003) 1560–1562, doi:10.1126/science.1077987.
- 657 [8] T. A. Scambos, J. A. Bohlander, S. C. A, P. Skvarca, Glacier ac-
658 celeration and thinning after ice shelf collapse in the the Larsen
659 B embayment, Antarctica, *Geophys. Res. Lett.* 31 (2004) L18402,
660 doi:10.1029/2004GL020670.

- 661 [9] E. Rignot, J. L. Bamber, M. R. Van den Broeke, C. Davis, Y. Li, W. J.
662 Van de Berg, E. Van Meijgaard, Recent Antarctic ice mass loss from
663 radar interferometry and regional climate modelling, *Nature Geoscience*
664 1 (2) (2008) 106–110.
- 665 [10] D. J. Wingham, D. W. Wallis, A. Shepherd, Spatial and temporal evo-
666 lution of Pine Island Glacier thinning, 1995–2006, *Geophys. Res. Lett.*
667 36 (17) (2009) L17501, doi:10.1029/2009GL039126.
- 668 [11] D. M. Holland, R. H. Thomas, B. de Young, M. H. Ribergaard, B. Ly-
669 berth, Acceleration of Jakobshavn Isbræ triggered by warm subsurface
670 ocean waters, *Nature Geoscience* 1 (2008) 659–664.
- 671 [12] M. Thoma, A. Jenkins, D. Holland, S. Jacobs, Modelling Cir-
672 cumpolar Deep Water intrusions on the Amundsen Sea continental
673 shelf, Antarctica, *Geophys. Res. Lett.* 35 (18) (2008) L18602, doi:
674 10.1029/2008GL034939.
- 675 [13] A. Shepherd, A. Hubbard, P. Nienow, M. King, M. McMillan, I. Joughin,
676 Greenland ice sheet motion coupled with daily melting in late summer,
677 *Geophys. Res. Lett.* 36 (1) (2009) L01501, doi:10.1029/2008GL035758.
- 678 [14] M. van den Broeke, J. Bamber, J. Ettema, E. Rignot, E. Schrama, W. J.
679 van de Berg, E. van Meijgaard, I. Velicogna, B. Wouters, Partitioning
680 recent Greenland mass loss, *Science* 326 (5955) (2009) 984–986, doi:
681 10.1126/science.1178176.
- 682 [15] C. Schoof, Ice sheet grounding line dynamics: steady states, sta-
683 bility, and hysteresis, *J. Geophys. Res.* 112 (2007) F03S28, doi:
684 10.1029/2006JF000664.
- 685 [16] R. F. Katz, M. G. Worster, Stability of ice-sheet grounding lines, *Proc.*
686 *R. Soc. A* 466 (2010) 1597–1620.
- 687 [17] D. Goldberg, D. M. Holland, C. Schoof, Grounding line movement and
688 ice shelf buttressing in marine ice sheets, *J. Geophys. Res.* 114 (F4)
689 (2009) F04026, doi:10.1029/2008JF001227.
- 690 [18] R. Alley, P. U. Clark, P. Huybrechts, I. Joughin, Ice-sheet
691 and sea-level changes, *Science* 310 (5747) (2005) 456–460, doi:
692 10.1126/science.1114613.

- 693 [19] J. Bamber, R. B. Alley, I. Joughin, Rapid response of modern day ice
694 sheets to external forcing, *Earth Planet. Sc. Lett.* 257 (1-2) (2007) 1–13,
695 doi:10.1016/j.epsl.2007.03.005.
- 696 [20] A. Shepherd, D. Wingham, Recent Sea-Level Contributions of the
697 Antarctic and Greenland Ice Sheets, *Science* 315 (5818) (2007) 1529–
698 1532, doi:10.1126/science.1136776.
- 699 [21] R. Alley, *Climate Change 2007: The Physical Science Basis. Contribu-*
700 *tion of Working Group I to the Intergovernmental Panel on Climate*
701 *Change, chapter 1: Summary for Policymakers.*, Cambridge University
702 Press, 2007.
- 703 [22] I. C. Rutt, M. Hagdorn, N. R. J. Hulton, A. J. Payne, The Glimmer
704 community ice sheet model, *J. Geophys. Res.* 114 (2009) F02004, doi:
705 10.1029/2008JF001015.
- 706 [23] W. Paterson, *The physics of glaciers*, Oxford, Pergamon, Tarrytown,
707 N.J., 1994.
- 708 [24] G. Durand, O. Gagliardini, B. Fleurian, T. Zwinger, E. Le Meur, Marine
709 ice sheet dynamics: Hysteresis and neutral equilibrium, *J. Geophys. Res.*
710 114 (2009) F03009, doi:10.1029/2008JF001170.
- 711 [25] F. Pattyn, A new three-dimensional higher-order thermomechanical ice
712 sheet model: Basic sensitivity, ice stream development, and ice flow
713 across subglacial lakes, *J. Geophys. Res.* 108 (B8) (2003) 2382, doi:
714 10.1029/2002JB002329.
- 715 [26] C. Schoof, R. C. A. Hindmarsh, Thin-film flows with wall slip: an asymp-
716 totic analysis of higher order glacier flow models, *Q. J. Mechanics Appl.*
717 *Math.* 63 (2010) 73–114, doi:10.1093/qjmam/hbp025.
- 718 [27] J. K. Dukowicz, S. F. Price, W. H. Lipscomb, Consistent approx-
719 imations and boundary conditions for ice-sheet dynamics from a
720 principle of least action, *J. Glaciol.* 56 (197) (2010) 480–496, doi:
721 10.3189/002214310792447851.
- 722 [28] T. Payne, S. Price, A three-dimensional, first-order model of ice flow:
723 numerical implementation and benchmarking. In preparation.

- 724 [29] F. Pattyn, L. Perichon, A. Aschwanden, B. Breuer, B. de Smedt,
725 O. Gagliardi, G. H. Gudmundsson, R. Hindmarsh, A. Hubbard, J. V.
726 Johnson, T. Kleiner, Y. Konovalov, C. Martin, A. J. Payne, D. Pollard,
727 S. Price, M. Rückamp, F. Saito, O. Souček, S. Sugiyama, T. Zwinger,
728 Benchmark experiments for higher-order and full Stokes ice sheet models
729 (ISMIP-HOM), *The Cryosphere Discuss.* 2 (2008) 111–151.
- 730 [30] R. Hindmarsh, A. Payne, Time-step limits for stable solutions of the
731 ice-sheet equation, *Ann. Glaciol.* 23 (1996) 74–85.
- 732 [31] J.-F. Lemieux, B. Tremblay, Numerical convergence of viscous-
733 plastic sea ice models, *J. Geophys. Res.* 114 (2009) C05009,
734 doi:10.1029/2008JC005017.
- 735 [32] C. Paniconi, M. Putti, A comparison of Picard and Newton iteration
736 in the numerical solution of multidimensional variably saturated flow
737 problems, *Water Resour. Res.* 30 (12) (1994) 3357–3374.
- 738 [33] J. M. Reisner, V. A. Mousseau, A. A. Wyszogrodzki, D. A. Knoll, An
739 implicitly balanced hurricane model with physics-based preconditioning,
740 *Mon. Wea. Rev.* 133 (2005) 1003–1022.
- 741 [34] K. J. Evans, D. A. Knoll, M. A. Pernice, Development of a 2-D algorithm
742 to simulate convection and phase transition efficiently, *J. Comput. Phys.*
743 219 (2006) 404–417, doi:10.1016/j.jcp.2006.03.025.
- 744 [35] J.-F. Lemieux, B. Tremblay, J. Sedláček, P. Tupper, S. Thomas,
745 D. Huard, J.-P. Auclair, Improving the numerical convergence of viscous-
746 plastic sea ice models with the Jacobian-free Newton Krylov method, *J.*
747 *Comput. Phys.* 229 (2010) 2840–2852, doi:10.1016/j.jcp.2009.12.011.
- 748 [36] M. Seager, A SLAP for the masses, Tech. Rep. UCRL-100267, Lawrence
749 Livermore Nat. Lab., Livermore, Calif. (1988).
- 750 [37] Y. Saad, M. H. Schultz, GMRES: a generalized minimal residual al-
751 gorithm for solving nonsymmetric linear systems, *SIAM J. Sci. Stat.*
752 *Comput.* 7 (3) (1986) 856–869.
- 753 [38] O. Axelsson, *Iterative Solution Methods*, Cambridge University Press,
754 New York, 1994.

- 755 [39] D. A. Knoll, D. E. Keyes, Jacobian-free Newton-Krylov methods: a
756 survey of approaches and applications, *J. Comput. Phys.* 193 (2) (2004)
757 357–397, doi:10.1016/j.jcp.2003.08.010.
- 758 [40] Y. Saad, *Iterative methods for sparse linear systems*, PWS, 1996.
- 759 [41] R. S. Tuminaro, H. F. Walker, J. N. Shadid, On backtrack-
760 ing failure in Newton-GMRES methods with a demonstration for
761 the Navier-Stokes equations, *J. Comput. Phys.* 180 (2002) 549–558,
762 doi:10.1006/jcph.2002.7102.
- 763 [42] S. C. Eisenstat, H. F. Walker, Choosing the forcing terms in an inexact
764 Newton method, *SIAM J. Sci. Comput.* 17 (1996) 16–32.
- 765 [43] J. L. Bamber, R. L. Layberry, S. P. Gogenini, A new ice thickness and
766 bed data set for the Greenland ice sheet 1: Measurement, data reduction,
767 and errors., *J. Geophys. Res.* 106 (D24) (2001) 33773–33780.
- 768 [44] J. K. Dukowicz, J. R. Baumgardner, Incremental remapping as a trans-
769 port/;advection algorithm, *J. Comput. Phys.* 160 (2000) 318–335, doi:
770 10.1006/jcph.2000.6465.
- 771 [45] W. H. Lipscomb, E. C. Hunke, Modeling sea ice transport using incre-
772 mental remapping, *Mon. Wea. Rev.* 132 (2004) 1341–1354.
- 773 [46] D. A. Knoll, W. J. Rider, A multigrid preconditioned Newton-Krylov
774 method, *SIAM J. Sci. Comput.* 21 (1999) 691–710.
- 775 [47] M. A. Heroux, R. A. Bartlett, V. E. Howe, R. J. Hoekstra, J. J. Hu,
776 T. G. Kolda, R. B. Lehoucq, K. R. Long, R. P. Pawlowski, E. T. Phipps,
777 A. G. Salinger, H. K. Thornquist, R. S. Tuminaro, J. M. Willenbring,
778 A. Williams, An overview of the Trilinos project, *ACM Trans. Math.*
779 *Soft.* 31 (3) (2005) 397–423.