Geosci. Model Dev., 9, 3907–3918, 2016
www.geosci-model-dev.net/9/3907/2016/
doi:10.5194/gmd-9-3907-2016

Geoscientific Model Development
Open Access

An approach to computing discrete adjoints for MPI-parallelized models applied to Ice Sheet System Model 4.11

Eric Larour[1], **Jean Utke**[2], **Anton Bovin**[3], **Mathieu Morlighem**[4], **and Gilberto Perez**[5]

[1]Jet Propulsion Laboratory, California Institute of technology, 4800 Oak Grove Drive MS 300-323, Pasadena, CA 91109-8099, USA
[2]Allstate Insurance Company, 2775 Sanders Rd., Northbrook, IL 60062, USA
[3]Department of Physics, The University of Chicago, 5720 South Ellis Avenue, Chicago, IL 60637, USA
[4]University of California, Irvine, Department of Earth System Science, Croul Hall, Irvine, CA 92697-3100, USA
[5]University of California, Irvine, School of Information and Computer Sciences, Irvine, CA 92697-3100, USA

Correspondence to: Eric Larour (eric.larour@jpl.nasa.gov)

Received: 26 April 2016 – Published in Geosci. Model Dev. Discuss.: 9 May 2016
Revised: 15 September 2016 – Accepted: 6 October 2016 – Published: 1 November 2016

Abstract. Within the framework of sea-level rise projections, there is a strong need for hindcast validation of the evolution of polar ice sheets in a way that tightly matches observational records (from radar, gravity, and altimetry observations mainly). However, the computational requirements for making hindcast reconstructions possible are severe and rely mainly on the evaluation of the adjoint state of transient ice-flow models. Here, we look at the computation of adjoints in the context of the NASA/JPL/UCI Ice Sheet System Model (ISSM), written in C++ and designed for parallel execution with MPI. We present the adaptations required in the way the software is designed and written, but also generic adaptations in the tools facilitating the adjoint computations. We concentrate on the use of operator overloading coupled with the AdjoinableMPI library to achieve the adjoint computation of the ISSM. We present a comprehensive approach to (1) carry out type changing through the ISSM, hence facilitating operator overloading, (2) bind to external solvers such as MUMPS and GSL-LU, and (3) handle MPI-based parallelism to scale the capability. We demonstrate the success of the approach by computing sensitivities of hindcast metrics such as the misfit to observed records of surface altimetry on the northeastern Greenland Ice Stream, or the misfit to observed records of surface velocities on Upernavik Glacier, central West Greenland. We also provide metrics for the scalability of the approach, and the expected performance. This approach has the potential to enable a new generation of hindcast-validated projections that make full use of the wealth of datasets currently being collected, or already collected, in Greenland and Antarctica.

1 Introduction

Constant monitoring of polar ice sheets through remote sensing, in particular since the advent of altimeter, radar, and gravity sensors such as ICESat-1, CryoSat, RADARSAT-1, ERS-1 and ERS-2, Envisat, and GRACE, has created a large amount of data that has yet to find its way through ice sheet models (ISMs) and hindcast reconstructions of polar ice sheet evolution. In particular, as evidenced by the wide discrepancy between ISMs involved in the SeaRISE and Ice2Sea projects (Nowicki et al., 2013; Bindschadler et al., 2013), significant improvements in modeled projections of the mass balance of polar ice sheets and their contribution to future sea-level rise have not resulted from the increase in availability of data, but rather from improvements in the type of physics captured in forward models. One reason for this is the lack of data assimilation capabilities embedded in the current generation of state-of-the-art ISMs. In the past 10 years, great strides have been made in improving model initialization by using steady-state model inversions of basal friction (MacAyeal, 1993; Morlighem et al., 2010; Larour et al., 2012; Price et al., 2011; Arthern and Gudmundsson, 2010), ice rheology (Rommelaere and MacAyeal, 1997; Larour et al., 2005; Khazendar et al., 2007), and bedrock elevation (Morlighem et al., 2014), among others. However, these approaches aim at improving our knowledge of poorly constrained input parameters and boundary conditions as long as the ice-flow regime is captured in a steady-state configuration. These inversions rely on analytically derived adjoint states of forward stress balance or mass-transport models, but do not extend to transient regimes of ice flow.

Applications to transient models and long temporal time series such as the ICESat/CryoSat continuous altimetry record from 2003 to the present day have been much more rare, and, to our knowledge, are limited to a few studies such as Heimbach and Bugnion (2009), Goldberg and Sergienko (2011), Goldberg and Heimbach (2013), Larour et al. (2014), and Goldberg et al. (2015), among others. The main issue here precluding widespread application of transient data assimilation lies in the difficulty in deriving temporal adjoints of transient models. In many cases, the sheer number of physical processes being represented makes the manual derivation of the adjoint state of a forward model extremely cumbersome and difficult to maintain. This state of affairs can be mitigated by adopting different approaches, such as (1) ensemble runs, as in Applegate et al. (2012), where model runs compatible with observations are selected; (2) methods similar to the flux-correction methods implemented in Aschwanden et al. (2013); Price et al. (2011), where boundary conditions are corrected in order to match time series of observations (tuning approach); (3) quasi-static approaches, where snapshot inversions are carried out in time, as in Habermann et al. (2012, 2013); and (4) sampling methods, which have the main drawback of being computationally very expensive (each sample at the cost of one forward run). Though this is not an exhaustive list of all available methods, the main advantage of adjoint-driven inversions is that it relies on the exact sensitivity of a forward model to its inputs, hence ensuring a physically meaningful inversion.

Understanding sensitivities of a forward ice-flow model, which is needed to physically constrain a temporal inversion, requires computation of derivatives of model outputs to model inputs. If such derivatives are approximated by finite-difference schemes, they are subject to the tradeoff between approximation and truncation errors for the perturbation, which is aggravated for higher-order derivatives. If the derivatives are computed using algorithmic differentiation (AD; Griewank and Walther, 2008), also known as automatic differentiation, then one can attain derivatives at a floating-point precision equivalent to that of the underlying program implementation of the numerical model, provided it is amenable to the application of an AD tool. In particular, this approach does not depend on the type of physics relied upon, and it is transparent to the model equations, provided each step of the overall software is differentiable. Indeed, the AD method assumes a numerical model M that is a function

$$\begin{array}{rccc} \boldsymbol{f}: & \mathbb{R}^n & \rightarrow & \mathbb{R}^m \\ & \boldsymbol{x} & \longmapsto & \boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x}) \end{array} \tag{1}$$

implemented as a computer program. The execution of the program implies the execution of a sequence of arithmetic operators such as $+$, $-$, and $*$ and intrinsics $\sin, e^x$, and so forth to which the chain rule can be applied. For each such elemental operation $r = \phi(a, b, \ldots)$ with result r and arguments $a, b, \ldots$[1] we can write the total derivative as

$$\dot{r} = \frac{\partial \phi}{\partial a}\dot{a} + \frac{\partial \phi}{\partial b}\dot{b} + \ldots. \tag{2}$$

For example, if ϕ is the multiplication operator $*$, i.e., $r = ab$, then one will get the product rule $\dot{r} = b\dot{a} + a\dot{b}$. Applying the above rule to each elemental operation in the sequence gives a method to compute

$$\dot{\boldsymbol{y}} = \boldsymbol{J}\dot{\boldsymbol{x}} \quad \text{with the Jacobian,}$$

$$\boldsymbol{J} = \left[\frac{\partial f_i}{\partial x_j}\right], i = 1\ldots m,\ j = 1\ldots n,$$

without explicitly forming the Jacobian. This method applies the chain rule in the computation order of the values in the program and is known as forward-mode AD. The opposite order of applying the chain rule to the elemental operations, known as reverse or adjoint-mode AD, yields projections $\overline{\boldsymbol{x}} = \boldsymbol{J}^T\overline{\boldsymbol{y}}$.

This is achieved using the partial derivatives of the forward operation ϕ as weights in redistributing the adjoints of the result r to the adjoints of the respective arguments a, b following the rule

$$\overline{a} = \overline{a} + \frac{\partial \phi}{\partial a}\overline{r};\ \ \overline{b} = \overline{b} + \frac{\partial \phi}{\partial b}\overline{r};\ \ \ldots\overline{r}; = 0. \tag{3}$$

Here $\overline{v}$ denotes the adjoint corresponding to variable v. The final zeroing of the result adjoint $\overline{r}$ corresponds to the notion that r had been assigned a new value overriding any previous value r might have had. Correspondingly the adjoint of r has been completely distributed, and to allow the increment formulation for the adjoint of any prior value of r, it has to be (re)set to 0.

For the multiplication example one therefore gets $\overline{a} = \overline{a} + b\overline{r};\ \ \overline{b} = \overline{b} + a\overline{r};\ \ \overline{r} = 0$. In particular, for applications in which $m \ll n$, the reverse mode is advantageous because its computational cost depends on m reverse sweeps instead of n forward sweeps. Typical problems in cryosphere science involve computations of diagnostics that are scalar-valued cost functions ($m = 1$), such as for example the spatio-temporally averaged misfit between modeled surface elevation and observed surface topography (Larour et al., 2014). For these

[1]In practice most ϕ are uni- or bi-variate.

cases, one can compute the gradient $\nabla f = \boldsymbol{J}^T \overline{y}$ with $\overline{y} = 1$ as a single projection (where f is the function defined in Eq. 1). Thus, for high-resolution models implying very large n, the reverse mode is an enabling and potentially very efficient technique. This significant capability in AD is what makes its application to data assimilation so efficient. Instead of evaluating a Jacobian using one forward sweep for each one of the model inputs, which would be significantly consuming as it scales in n, the reverse mode evaluates the gradient of one specific output of interest with respect to the model inputs in only one reverse sweep.

Using this approach, it is possible, without manual derivation, to evaluate the derivative $\frac{\partial f}{\partial x}$ from Eq. (1). In cryosphere models, f could typically be a transient ice-flow model, with inputs x taken for example as the basal drag coefficient α at the ice–bed interface, and the output y as the cost function measuring the spatio-temporally averaged difference between for example surface elevation and an altimetry record. In this case, $m = 1$, and n is the number of degrees of freedom of the basal drag input. Obtaining the sensitivity function $\frac{\partial f}{\partial x}$ is critical both in terms of enabling data assimilation using inversion methods and in terms of understanding the sensitivity of a model to its model inputs, as well as the impact of processes represented in the model itself. Approaches other than AD can be relied upon to compute such a sensitivity, such as solving for the adjoint equation directly (Morlighem et al., 2010; Larour et al., 2012; Cornford et al., 2013), but usually, a mix of forward AD or reverse AD is implemented for formulations where the manual derivation is difficult (Pawlowski et al., 2012; Goldberg and Heimbach, 2013). In Pawlowski et al. (2012) for example, AD relies on template-based generic programming to compute sensitivities, while in Goldberg and Heimbach (2013), AD is based on source-to-source transformation. Our approach will instead be based on so-called operator overloading, which is uniquely suited to a framework written in C++.

Applying AD to large-scale frameworks such as the Ice Sheet System Model (ISSM) (Larour et al., 2012), MITgm (Heimbach, 2008), SICOPOLIS (Heimbach and Bugnion, 2009), or DassFlow (Monnier, 2010, among others) is a difficult proposition but one that enables significant improvements in the way models can be initialized (Heimbach and Bugnion, 2009), hindcast validated (Larour et al., 2012), and calibrated (Goldberg et al., 2015) towards better projections. Traditional approaches relying on source-to-source transformation have been developed, but for frameworks such as the ISSM, which are C++ oriented, and highly parallelized, this type of approach breaks down. Our goal here is to demonstrate how the so-called operator-overloading AD approach can be implemented and validated for a framework such as the ISSM, and what developments were necessary to make this capability operational. Our approach is discussed in Sect. 2 of this paper, with Sect. 3 describing the method validation as well as applications with the ISSM. We discuss and conclude in the last section the applicability of such an approach to other frameworks, and the opportunities these new developments afford for cryosphere science and data assimilation of remote sensing data in particular.

2 Methodology and validation

2.1 Source-to-source vs. overloaded operators

Distinctions between the AD implementation approaches relate to the way the derivative data in $\dot{v}$ or $\overline{v}$, respectively, are associated with the original program variable v (data augmentation) and how the logic for coefficient propagation is added to the original program (logic augmentation). The basic options for the former are association by address and association by name. Association by address packs the original and the derivative data into a new type called the active type, and all differentiated program variables have their type changed to that new active type. For the logic augmentation one uses source code transformation or operator overloading. Because of the complexity of the C++ syntax and semantics, there currently still is no comprehensive AD tool for source code transformation of C++ models, and thus operator overloading remains the method of choice, implying association by address for the data augmentation. In practice, the overloaded operators and intrinsics for the forward mode typically execute Eq. (2) directly for the forward mode and for the reverse mode record the sequence of operations into a trace $T(f(\boldsymbol{x}))$, which then is read backward by an interpreter $R(T)$ that executes the statements Eq. (3) for the reverse mode. The invocation of $R(T)$ is part of the user-written logic that then uses the derivatives. The model thus enhanced by an adjoint capability computing both f and ∇f is denoted by $\overline{M}$.

2.2 Type change to enable overloaded operators

Changing to the aforementioned active type is a significant effort to be undertaken in the model code. Among the choices to effect this type change, one should select one that is transparent to the model development process, is maintainable, and minimizes the manual effort.

The tool of choice for this paper is ADOL-C (Griewank et al., 1996; ADOL-C, 2016) and our target model M is the ISSM (Larour et al., 2012). The ADOL-C manual is quiet on the practical approach to effect the type change. For models with a large C++ code base and many contributing developers such as the ISSM, not all of them aware of $\overline{M}$ but merely interested in running M, it is important to make this process transparent and robust. The representation of M as a computer program typically means that in the set of program variables $\mathcal{V} = \mathcal{A} \cup \mathcal{P}$ there is a subset $\mathcal{P}$ of passive variables that do not carry derivative information, such as variables of non-differentiable type (integers, strings) but also floating-point parameters, physical constants, or descriptors of the problem domain. The size of the trace T is a major factor in the computational efficiency of the adjoint. Thus, one must categorize program variables into $\mathcal{A}$ and $\mathcal{P}$ with the aim of minimizing the set $\mathcal{A}$ of active (i.e., type-changed) program variables. In the ISSM this is accomplished by changing `double` variables to a type named for brevity, e.g., `TA` (in the ISSM `IssmDouble`) for variables in $\mathcal{A}$, and to, e.g., `TP` (in the ISSM `IssmPDouble`) for variables in $\mathcal{P}$, respectively. This categorization must be performed by an expert familiar with the global data dependencies in M and the work that has been done for $\overline{M}$. In addition, it can be incrementally done for code contributions supplied as `double` variables without breaking M. This is a key aspect of the overloaded AD approach that is amenable to easy maintenance and expansion of a complex C++ framework such as the ISSM, and one of the key reasons why this approach was selected to compute $\overline{M}$ in the ISSM. In the end, all `double` types should eventually be replaced with either `TA` or `TP`. Following good practices in C++ both `TA` and `TP` are `typedefed` in a central location switched only there via a preprocessing macro to use the active type supplied by the AD tool or hide the distinction altogether for running the plain M. This implies that one cannot explicitly overload methods or define data structures for distinct `TA` and `TP` without filtering the declaration and definition of the `TA` variant by the preprocessor as well. Such code duplication introduces an undue code maintenance burden.

Instead, the recommended approach is the use of C++ template classes and methods where `TA` and `TP` are the concrete arguments for an abstract template type `T`. While the use of templates may imply a larger up-front effort if they were not present in the model code before, as was the case in the ISSM, the long-term benefits not only for $\overline{M}$ but also the plain M are obvious when one wants to experiment with computation at different precision levels. Aside from the ISSM, an established general example of this kind of templating is the use of SACADO within Trilinos (The Trilinos Project, 2014; Phipps et al., 2008). Of particular value in the ISSM was the migration of data containers for arrays and (sparse) matrices and that of existing wrappers to `malloc` and `free` to templated wrappers of `new` and `delete`, called `xNew` and `xDelete` in the ISSM. The latter is necessary not only to properly instantiate `TA` variables, but also permits specializations for `TA` in $\overline{M}$ as explained in the following.

Overall, the use of C++ template classes introduces an additional element of complexity to the code, but it also means that the same exact framework can be used to compute both the forward run and its gradient, using the exact same code base, without the need to manage different code for the computation of $\overline{M}$. This in turn means that the ISSM can be compiled using ADOLC and shipped to the community in only one binary version that can accommodate both M and $\overline{M}$, which is a critical feature in terms of extending AD features to the wider community without having to dive into the intrinsic complexities of AD transformation itself.

Interpreting the trace by $R(T)$ means that the space holding the data for the variables $r, a, b \in \mathcal{V}^*$ (the set of instantiated program variables at runtime) occurring in each operation $r = \phi(a, b)$ must be represented by some mapping $\omega : \mathcal{V}^* \longmapsto \mathbb{N}^+$ to pseudo addresses. In ADOL-C these addresses are called locations and represent indices in a work array held by $R(T)$. The pseudo addresses must be managed through the `TA` constructor and destructor in a fashion similar to the memory management of the actual program variables themselves; i.e., pseudo addresses are assigned from and returned to a pool Ω of available addresses. However, no distinction between heap and stack variables is made and generally data locality will not be preserved. On the other hand, for special operations ϕ_A with array arguments of size s, that is, calls to external solvers (see Sect. 2.3) or MPI routines (see Sect. 2.4), it would be counterproductive to record in T the pseudo addresses of each of the s array elements rather than a consecutive range. The latter, however, requires that the pool be primed by some call $\Omega(s)$ such that ω returns consecutive pseudo addresses when called by the constructor for each element in the array. In ADOL-C this is done by calling `ensureContiguousLocations` immediately before an active array is instantiated. Avoiding copying of array values in M as an efficiency measure means that any given array has a good chance of being used in ϕ_A, and to avoid littering the code with preprocessor-guarded calls to $\Omega(s)$, we decided in the ISSM to instead add a call to `ensureContiguousLocations` in the `TA` specialization of `xNew`:

```
#if defined(_HAVE_ADOL_C_)
template <>
adouble* xNew(unsigned int s) {
 ensureContiguousLocations(s);
 adouble* aT_p=new adouble[s];
 assert(aT_p);
 return aT_p;
}
#endif
```

Thus, the use of ϕ_A means frequent calls to $\Omega(s)$. The implementation of $\Omega(s)$ searches the pool to find a sufficiently large consecutive range or otherwise grow the pool by at least s addresses. In the ISSM, this search became a significant performance bottleneck. Minimizing the search effort and balancing it with pool growth has been newly implemented in ADOL-C and is controlled by setting parameters via `setStoreManagerControl` for the ratio of the addresses inside and outside of the pool and the maximal pool size to trigger searches.

2.3 Binding to external solvers

The strength of the AD approach lies in the automated differentiation of all the parts of the model that are changing and expanding as the model is being developed, as opposed to manually programming adjoints that are error-prone, and implies a significant effort in terms of code development and maintenance. However, because models reference libraries, for fixed mathematical mappings that may have easily programmable high-level adjoints, one can and should exploit this insight as it can yield efficiency gains not obtainable by an AD tool remaining unaware that calls to some collection of subroutines represent a certain mathematical mapping. A good example are linear solver libraries that may not even be written in the same programming language as the model in question. Considering the system $As = b$, the solver S computing $s = S(A, b)$ needs an adjoint counterpart solving $A^T t = \overline{s}$, incrementing $\overline{b} = \overline{b} + t$ and, if $A \in \mathcal{A}$, also $\overline{A} = \overline{A} - \overline{b}s^T$. Clearly, the solve with the transposed can easily be implemented reusing the solver S as in $t = S(A^T, \overline{s})$. However, the answers to the following questions affect the efficiency and ought to be considered in the context of the given model (Giles, 2008).

Q1: Is $A \in \mathcal{A}$, which therefore requires the rank-1 update to $\overline{A}$?

Q2: Are the previous values of s, r, and A overwritten by S but used to compute partial derivatives for Eq. (3), and therefore must be saved and restored?

Q3: If S is a direct solver, should one save the factors or refactorize for the adjoint solve?

Q4: How does the external function interface used by $R(T)$ allow for efficient reuse of intermediate buffers?

External functions f_e that had been supported by ADOL-C had the signature $f_\text{e}(l_x, x, l_y, y)$ with inputs x, outputs y for the original call, l_x, l_y for their respective array lengths, and $\overline{f_\text{e}}(l_y, \overline{y}, l_x, \overline{x})$ for the adjoint counterpart. In the case of a linear system with $A \in \mathcal{A}$, the input x is packed with both A and r, while y contains the solution s of the system on return. This, however, was insufficient for binding to solvers from the GNU Scientific Library (GSL; Galassi, 2009) and MUMPS (Amestoy et al., 2001) used by the ISSM. The following extensions were implemented in ADOL-C to enable the ISSM adjoints, but are generic in nature and would need to be supported in some form by other tools.

E1: expanded the $\overline{f_\text{e}}$ interface by x, y to $\overline{f_\text{e}}(l_y, \overline{y}, l_x, \overline{x}, x, y)$ to enable refactoring;

E2: added optional parameters to pass in the sparsity pattern of A for MUMPS as a generic integer array i of length l_i for both f_e and $\overline{f_\text{e}}$;

E3: added controls for storing and restoring prior values of x and y;

E4: added tracking of the maximal l_x, l_y in sequences of f_e calls.

Regarding Q1, we know that in the ISSM, $A \in \mathcal{A}$, and regarding Q2, the parameters passed to f_e have no other uses and, therefore, using the controls (E3), we avoid (re)storing their values. The direct solver from the GSL used here had no API control to back-solve with the factors for the transposed and we did not want to reverse engineer the permutation representation. Hence the refactoring was done as a matter of convenience for the sequential reference case requiring E1. The parallel and therefore practically more efficient MUMPS solver operates on sparse, distributed A, therefore requiring E2. MUMPS offers both the ability to store the factors to file and perform the back-solve for the transpose. However, the MUMPS portion of the runtime is comparatively small (see Sect. 3). Consequently the overhead for the file I/O when considering the factor data size after fill-in is not expected to yield much practical benefit in this context, answering Q3. Finally, for Q4, extension E4 is exploited because transient runs of the ISSM need to account for changes in the system size, and a preallocation with the maximal buffer sizes therefore avoids some of the memory management overhead.

2.4 Handling parallelism with the AdjoinableMPI library

As is the case with the ISSM, practically relevant science problems incur a computational complexity that necessitates execution on parallel hardware, often using MPI as the paradigm of choice. Sending data with MPI from a source buffer s to a target buffer t can be interpreted as a simple assignment $t = s$. This implies for the adjoint an increment $\overline{s} = \overline{s} + \overline{t}$; that is, the adjoint of a communication is a reversal of the data flow between buffers. Calculating the adjoint of $t = s$ means sending back the value of $\overline{t}$ and computing $\overline{s} = \overline{s} + \overline{t}$. The principles for adjoining two-sided MPI communication have been explored in Utke et al. (2009). The development of the AdjoinableMPI (AMPI) wrapper library AdjoinableMPI (2016) started in the fall of 2012. It is designed to provide an AD tool-independent implementation for adjoining MPI-parallelized C, C++, and Fortran models. The wrapper interfaces distinguish themselves from the original MPI by the prefix `AMPI` and have a few additional parameters where needed to enable the adjoint functionality. AMPI also provides additional types and predefined symbols. Discussing the internal design of AMPI is outside the scope of this paper. However, the application to the ISSM is the first large-scale practical use of AMPI and in the following we will discuss the steps taken to use it in the ISSM code base.

For testing and small-scale experiments, the ISSM code, as many other models, treats MPI parallelization as a compile-time option controlled by preprocessor macros. Furthermore, turning the adjoint capability on and off as suggested in Sect. 2.2 would imply additional switching between the original MPI and AMPI with code duplication and the potential for errors. To avoid these undesirable consequences we de-

Table 1. Logic variants encapsulated in the ISSM MPI wrapper library.

	MPI	AD	
1	no	no	emulate MPI semantic, e.g., with `memcpy` between buffers for `MPI_Reduce`
2	no	yes	emulate MPI semantic, but for differentiation need to use the overloaded assignments instead of `memcpy`
3	yes	no	pass-through to plain MPI
4	yes	yes	call AMPI routines

cided to introduce in the ISSM another wrapper layer (prefixed `ISSM_MPI`) of calls and definitions to encapsulate completely the four functionality variants listed in Table 1.

The approach is shown for `MPI_Reduce` in Fig. 1.

Variant 1 is implemented by line 21, variant 2 by lines 13–18 and 21, variant 3 by line 9, and variant 4 by line 7. The example code reflects some of the simplifying assumptions being made based on the MPI usage patterns in the ISSM. Examples are the absence of `MPI_IN_PLACE` and user-defined MPI types as well as all active data being of double precision. This permits the simple distinction in line 12, but of course the logic covers all other non-differentiated MPI types that may occur. The following lines (14, 15) indicate the pairing between the C++ type definition `IssmDouble` and the wrapper-defined `ISSM_MPI_DOUBLE` to signify active data being communicated when AD is enabled. Mirroring in MPI the approach described in Sect. 2.2 with two types `TA` and `TP`, the passive type `TP` (concrete `IssmPDouble`) has an MPI counterpart defined in the wrapper as `ISSM_MPI_PDOUBLE`. It too can be the value passed in via the `datatype` argument at line 1 and signals to AMPI passive floating-point communications. The definitions provided by the `ISSM_MPI` wrapper corresponding to types `TA` and `TP` are given in Table 2.

The matching between the actual type of the buffer and the corresponding MPI type must extend to the templating approach suggested for the type change in Sect. 2.2. The main reason is to avoid type errors, which become more common as the complexity of a code increases. Because the MPI standard keeps the definition of `MPI_Datatype` opaque and MPI types can be created at runtime, the value of `MPI_Datatype` cannot be used as the value template parameter itself (as it may not be a compile time constant). Therefore, a template buffer type `T` must be paired with the corresponding MPI type value declared as a pointer parameter, as shown in Fig. 2. Assuming many AMPI calls in `foo`, this confines the code locations where type errors may be introduced to the template instantiations themselves.

Finally, a practical concern for using the parallelized adjoint is the handling of sensitivities to quantities that are uniformly initialized across ranks, such as parameters. Frequently, as was the case within the ISSM, these quantities are initialized from files or otherwise per process in the parallel case the same way as in the sequential case. In the parallel case that implies a replication of the same quantity across ranks. However, to obtain the correct sensitivities, the quantity q in question should be unique; in other words, that quantity must be uniquely initialized at one root rank and then broadcast to the other ranks. Otherwise, for r ranks, at each rank one would then obtain only a part $\overline{q}_i$ of the total $\overline{q}$ and would have to "manually" sum up $\overline{q} = \sum \overline{q}_i$. With an initial broadcast of q, however, the corresponding adjoint provided through AMPI by using `AMPI_Bcast` is that exact sum reduction, and $R(T)$ yields the correct adjoint at the broadcast root. This notion similarly applies to any situation where a conceptually unique quantity of active type is implicitly replicated on some ranks.

2.5 Validation

The ISSM is validated in AD mode by continuously running a test suite within the Jenkins (Smart, 2011) integration and development framework (available here at http://issm.jpl.nasa.gov/developers/). A detailed description of the suite of benchmarks is given in Table 3. The aim is to (1) compare forward runs in the ISSM with their counterparts when overloaded operators are switched on (the results should be identical within double-precision tolerances); and to (2) compare forward and reverse runs carried out with the ISSM AD on and off, using the GSL and MUMPS solvers. Comparisons of gradients computed in AD mode with standard forward difference methods are also carried out to make sure the gradients computed (essentially in reverse scalar mode, which is the mode of predilection in the ISSM for data assimilation) in AD mode are accurate.

3 Application

3.1 Application

The ongoing use of adjoint computations includes sensitivity studies and state estimation problems for transient model runs. Because this paper concentrates on technical aspects, we show, only as exemplary evidence of the practical usefulness, some sensitivities of cost functions calibrated for two sensors commonly encountered in cryosphere science, altimeters (that measure surface elevation) and radars (that measure surface displacement or velocity). In Fig. 3, the sensitivity (gradient) of a cost function related to ICESat-1 altimetry with respect to surface mass balance (SMB) is shown

```
int ISSM_MPI_Reduce(void *sendbuf, void *recvbuf, int count,
                 ISSM_MPI_Datatype datatype, ISSM_MPI_Op op, int root,
                 ISSM_MPI_Comm comm){
        int rc=0;
 #ifdef _HAVE_MPI_
 #ifdef _HAVE_AMPI_
        rc=AMPI_Reduce(sendbuf, recvbuf, count, datatype, op, root, comm);
 #else
        rc=MPI_Reduce(sendbuf, recvbuf, count, datatype, op, root, comm);
 #endif
 #else
 #ifdef _HAVE_ADOLC_
        if (datatype==ISSM_MPI_DOUBLE) {
           IssmDouble* activeSendBuf=(IssmDouble*)sendbuf;
           IssmDouble* activeRecvBuf=(IssmDouble*)recvbuf;
           for(int i=0;i<count;++i) activeRecvBuf[i]=activeSendBuf[i];
        }
        else
 #endif
             memcpy(recvbuf,sendbuf,sizeHelper(datatype)*count);
 #endif
    return rc;
}
```

Figure 1. Code for wrapping reduction.

```
// this declaration comes from the wrapper:
extern ISSM_MPI_Datatype ourISSM_MPI_DOUBLEVal,ourISSM_MPI_DOUBLEVal;

template <class T, ISSM_MPI_Datatype *mt_p> void foo(T *t){
  ISSM_MPI_Reduce(t, ..., ,*mt_p,..);
}

void bar(IssmDouble *x, IssmPDouble *p) {
  foo<IssmDouble,&ourISSM_MPI_DOUBLEVal>(x);
  foo<IssmPDouble,&ourISSM_MPI_PDOUBLEVal>(p);
}
```

Figure 2. Code snippets for templating with corresponding MPI data types.

for three epochs, September 2003, June 2006, and February 2009. The cost function $J_{\mathrm{Altimetry}}$ is the spatio-temporal average of the misfit between modeled surface elevations (from the mass-transport module of the transient solution in the ISSM) and the corresponding ICESat-1 altimetry record. This cost function was used within an inversion method to reconstruct SMB over the entire ICESat-1 time record (Larour et al., 2014). This example would correspond to the case where function f from Eq. (1) is a transient ice-flow model (including mass transport and stress balance, not thermal, described in Larour et al., 2014), fully nonlinear in its treatment of the material rheology, and highly resolved both spatially (kilometer resolution at the coastline) and temporally (2-week time step). The final output (corresponding to y in Eq. 1) of the model is the altimetry cost function $J_{\mathrm{Altimetry}}$ itself and the model input SMB (corresponding to x in Eq. 1). The sensitivity displayed in Fig.3 is therefore the AD com-

Table 2. Types per variant from Table 1.

	`ISSM_MPI_DOUBLE` `#define` to	`IssmDouble` (= `TA`) `typedef` to	`ISSM_MPI_PDOUBLE` `#define` to	`IssmPDouble` (=`TP`) `typedef` to
1	2	`double`	3	`double`
2	2	`adouble`[1]	3	`double`
3	`MPI_DOUBLE`	`double`	`MPI_DOUBLE`	`double`
4	`AMPI_ADOUBLE`[2]	`adouble`[1]	`MPI_DOUBLE`	`double`

[1] Active type from ADOL-C. [2] Active MPI type from AMPI.

Table 3. ISSM AD validation suite integrated within Jenkins for continuous integration and delivery (Smart, 2011). Tests 3001 to 3010 and 3101 to 3110 test the repeatability of forward runs with and without ADOL-C compiled, but with no AD drivers specifically called. The forward runs involved are the standard stress balance, mass transport, and thermal solutions, with 2-D SSA (shelfy-stream approximation MacAyeal, 1989), 3-D SSA, 3-D HO (higher-order, Blatter, 1995; Pattyn, 1996), 3-D full-Stokes (Stokes, 1845), and DG (discontinuous-Galerkin) formulations. Test 3015 tests the AD GSL capability by comparing the AD forward scalar mode (where we compute the gradient of ice volume with respect to ice thickness at one vertex of the mesh) against a simple forward differences computation on the same gradient. Test 3019 validates the AD GSL capability in reverse scalar mode (where we compute ice volume gradient with respect to thickness at all vertices of the mesh) vs. the forward vectorial mode. Both gradients should be identical. Finally, test3119 validates the parallel capabilities (AD MUMPS) by comparing reverse scalar computations with GSL and MUMPS.

Test	Solution sequence	Formulation	Solver	Description
3001/3101	Stress balance	2-D SSA	GSL/MUMPS	Equal runs with overload on and off
3002/3102	Stress balance	3-D SSA	GSL/MUMPS	Equal runs with overload on and off
3003/3103	Stress balance	3-D HO	GSL/MUMPS	Equal runs with overload on and off
3004/3104	Stress balance	3-D FS	GSL/MUMPS	Equal runs with overload on and off
3005/3105	Mass transport	2-D	GSL/MUMPS	Equal runs with overload on and off
3006/3106	Mass transport	2-D (DG)	GSL/MUMPS	Equal runs with overload on and off
3007/3107	Mass transport	3-D	GSL/MUMPS	Equal runs with overload on and off
3008/3108	Thermal steady state	3-D	GSL/MUMPS	Equal runs with overload on and off
3009/3109	Thermal transient	3-D	GSL/MUMPS	Equal runs with overload on and off
3010/3110	Transient 2-D	2-D	GSL/MUMPS	Equal runs with overload on and off
3015	Mass transport 2-D	2-D	GSL	AD forward scalar vs. forward differences
3019	Thermal transient	3-D	GSL	AD reverse scalar vs. AD forward vectorial
3119	Thermal transient	3-D	MUMPS vs. GSL	AD reverse scalar (MUMPS vs. GSL) scalar

puted $\frac{\partial J_{\text{Altimetry}}}{\partial \text{SMB}}$ at different time steps corresponding to the transient model input SMB.

A second type of observation garnering considerable interest is temporally resolved time series of radar observations (using speckle-tracking or InSAR to infer surface deformation of the ice) to measure variations of surface velocity on a seasonal timescale (one observation every 2 months, as in Moon et al., 2015). Temporally inverting for such time series, trying to reconstruct for example basal friction, is a topic of interest due to its relevance in terms of understanding the dynamics of calving, basal slip, and shear softening, and associated feedback mechanisms. In Fig. 4, we demonstrate the feasibility of such inversions by computing the gradient of a cost function J_{Velocity} related to the Moon et al. (2015) time series with respect to basal drag α at epochs 2008, 2010, and 2013. The cost function is the spatio-temporal average of the misfit between modeled surface velocity (from the stress balance module of the transient solution in the ISSM) and observed surface velocities. This example would correspond to the case where function f from Eq. (1) is the same transient ice-flow model described previously, and the final output (corresponding to y in Eq. 1) of the model is the velocity cost function J_{Velocity} itself and the model input basal friction α (corresponding to x in Eq. 1). The sensitivity displayed in Fig. 4 is therefore the AD computed $\frac{\partial J_{\text{Velocity}}}{\partial \alpha}$ at different time steps corresponding to the transient model input α.

3.2 Benchmarking

Currently, these adjoint computations are performed on the Pleiades cluster at the NASA Advanced Supercomputing Center. Given the fact that the effort for computing the adjoint ∇f relies on one reverse sweep, the practical question becomes what the actual runtime overhead of computing both f and ∇f when compared to computing just the original f is. Here in particular one considers the effects of compiler optimization on the original code for f using the built-in floating-point operations on one side and in contrast

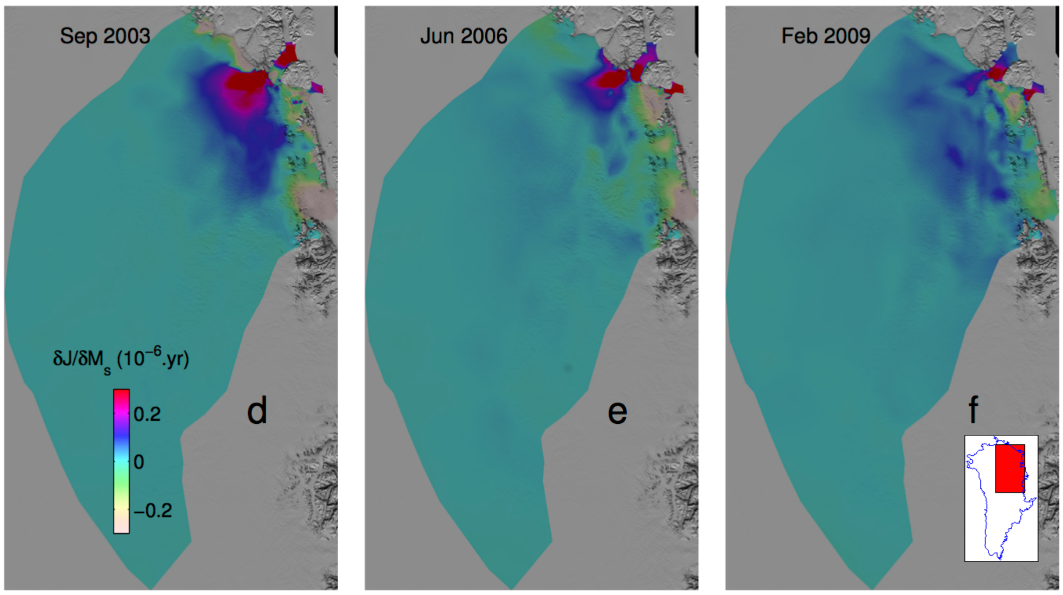

Figure 3. Gradient of the surface altimetry cost function (spatio-temporal average of the misfit between the 6-year record of observed ICESat altimetry and modeled surface elevation) with respect to SMB (in years). The gradient is computed for every time step of a 2-week interval time series between 2003 and 2009. We only show three periods (September 2003, June 2006, and February 2009) in the entire interval. The location for the study is the northeastern Greenland Ice Stream, and the gradient is taken from an inversion study of the surface forcings necessary to best fit the ICESat altimetry record (Larour et al., 2014).

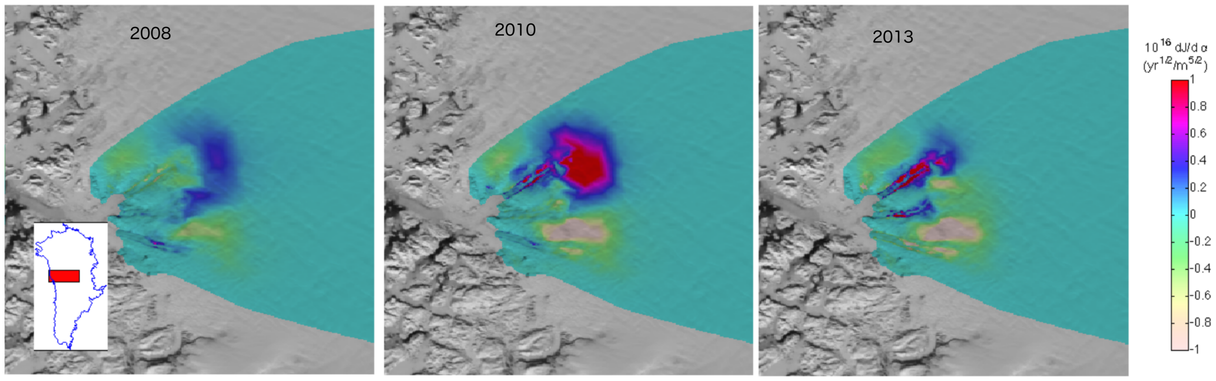

Figure 4. Gradient of the surface velocity cost function (spatio-temporal average of the misfit between a 5-year time series of observed surface velocities and the modeled surface velocity) with respect to the basal drag coefficient (in $\mathrm{yr}^{1/2}\,\mathrm{m}^{-5/2}$). The gradient is computed for every time step of a 2-week interval time series between 2008 and 2013. We only show three periods (2008, 2010, and 2013) in the entire interval. The location for the study is Upernavik Glacier, central West Greenland, and the gradient is taken from an inversion study of the basal forcings required to match observed RADAR time series of surface velocities and calving front positions.

to that the overhead incurred by calling the overloaded operators as well as the creation, storage, and interpretation of the trace T (as the means to compute ∇f) on the other. Aside from the fact that the tuning of the ISSM adjoint is a work in progress, we want to highlight the overwhelming impact of the application-specific aspects on the runtime ratio. Therefore, we emphasize that exhibiting any particular overhead number as the ultimate result of scenario-specific tuning is of little use to the practitioner wanting to answer science questions. Rather, using examples from the ISSM work, we want to show what may prevent us from achieving a satisfactory overhead.

The earliest ISSM adjoint computations took place before the MPI wrapper library was introduced and therefore were done sequentially with the GSL solvers. To evaluate the performance, a representative test case was chosen from the ISSM regression test suite, test 101. This test models the steady-state stress balance of a square ice shelf of size 50 km × 50 km and thickness 1000 m (at the grounding line) to 300 m (at the ice front). The ice shelf is clamped

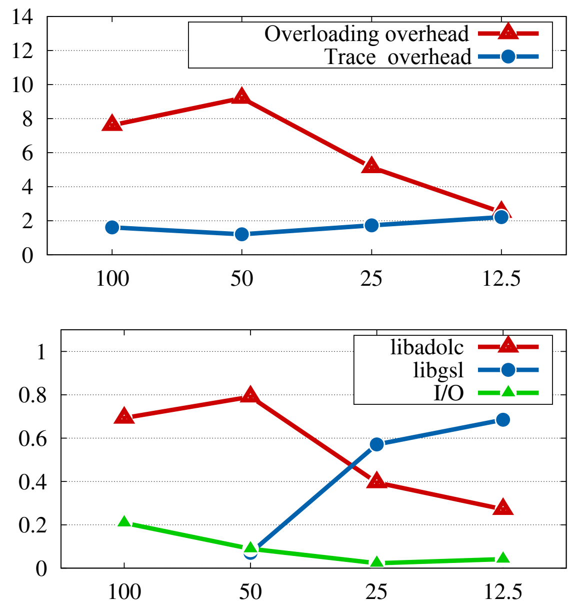

Figure 5. Test problem setup with the sequential GSL solver; plots over max mesh distance show overhead factors (upper frame) and approximate runtime portions of the model execution with adjoint computation (lower frame).

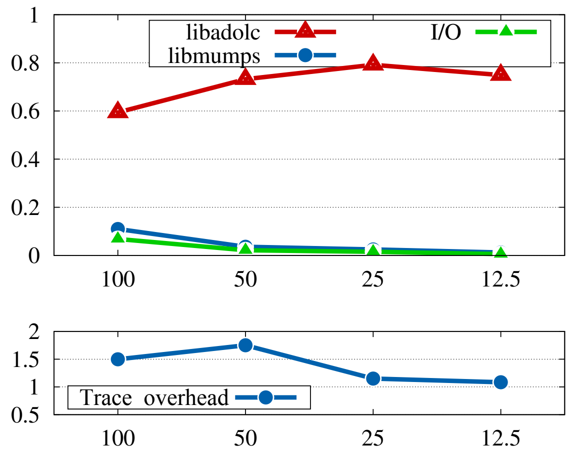

Figure 6. Test problem setup with the parallel MUMPS solver; plots over max mesh distance show approximate runtime portions of the model execution with adjoint computation (upper frame) and overhead factors for the trace creation and interpretation (lower frame).

at the grounding line, free to flow at the ice front (on one side only), with nonlinear viscosity dependent on Glen's law (Glen, 1955, 1958). The formulation relies on the shelfy-stream approximation (SSA, MacAyeal, 1989), and is inherently nonlinear, relying on a Picard iteration for convergence. For more details on this test, we refer the reader to Larour et al. (2012). For benchmarking, the problem size was indirectly set by specifying a measure for the maximal resolution of the generated mesh, thus increasing the number of mesh elements generated for a smaller-resolution parameter. The parallel runs were carried out using eight CPUs. The overhead factors are shown separately for the overloading as such and the generation of T and $R(T)$ as wall-clock comparisons to the unmodified model compiled with default optimization ($-O_2$) in Fig. 5 (upper frame). While this plot indicates a small overhead factor of ≈ 4.5 in particular for the largest mesh case (distance 12.5 km), the reason for this becomes apparent in the plot in the lower frame. It shows that the majority of the runtime is consumed by the GSL solver (libgsl), completely overshadowing any of the overhead caused by the adjoint for the largest mesh. We want to emphasize that GSL was chosen not for its efficiency, but for the simplicity of the setup, which quickly enabled adjoint computations. Introducing AMPI and thereby moving to a more appropriate solver (MUMPS) causes the adjoint overhead to become more prominent. The most consequential change necessitated by the use of MPI is the forced contiguity of the pseudo addresses (see Sect. 2.2).

In combination with the much reduced impact of the solver, the overhead factor for equivalent test problems reached temporarily up to 145, which clearly was not acceptable. Subsequent changes to the ADOL-C tool included improvements in the internal address management. Changes to the model included modifications of the sparse data format, enabling the control of the I/O buffering for the trace T and the ADOL-C address manager via the ISSM configuration specific to the setup to be computed. The combination of these changes led to regaining of better performance and effective overall overheads between 10 and 30. Analyzing the details of the performance shows that the overhead factor for the trace creation and interpretation does not change significantly (see Fig. 6, bottom) with the mesh size, in accordance with the theoretical result for the adjoint computation. The large majority of the total overhead, evidenced by the runtime portion for libadolc in Fig. 6 (top), still originates in the internal address management. While the overall overhead factors are sufficiently small for the practical use of the adjoints for science problems, further improvements in the addressing scheme are clearly warranted and the subject of ongoing work.

4 Conclusions

We developed a new adjoint capability in the ISSM, based on the ADOL-C framework and the AdjoinableMPI library, which is to our knowledge the first time this type of approach has been systematically applied to a software framework of this size and complexity. Despite the difficulties encountered in rewriting the software, the overloaded approach is transparent to the user, which is critical given the size of the larger cryosphere science community that is not familiar with the adjoint work, and for which classic approaches such as source-to-source transformation may not be applicable or may prove more cumbersome. The flexibility of this approach allows in particular for quick turn-around in developing adjoint models of new parameterizations that are not easily hand-derived. This is a major advantage in that it opens this approach to the wider community. This, given the large amount of remote sensing data currently being collected and under-utilized, could prove paramount if we are to hindcast validate projections of sea-level rise. Further work is of course required to bring in additional observations such as gravity sensors, or radar stratigraphy observations, which will involve development of new cost functions, and scalability in 3-D. Though this is complex in that it requires integrated resiliency and adjoint checkpointing schemes for long-running transient modeling scenarios, our approach has proven flexible, and should lead to a brand new set of data assimilation capabilities that have already been available to other Earth science communities for a long time. Indeed, by allowing temporal data assimilation for a large number of sensors and models, such as demonstrated here with the use of altimetry and radar sensors for mass transport and stress balance models, respectively, the ISSM paves the way for wider integration between the modeling and observational cryosphere communities.

5 Code availability

The ISSM code and its AD components are available at http://issm.jpl.nasa.gov. The instructions for the compilation of the ISSM in AD mode, along with test cases, are presented in the Supplement attached to this paper.

The Supplement related to this article is available online at doi:10.5194/gmd-9-3907-2016-supplement.

Acknowledgements. DE-AC02-06CH11357. Eric Larour and Jean Utke were supported by the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the NASA Cryospheric Sciences and Modeling and Analysis Programs. Gilberto Perez was supported by a subcontract from the Jet Propulsion Laboratory to the University of California at Irvine and Mathieu Morlighem was supported under a contract with the NASA IceBridge research program.

Edited by: D. Goldberg
Reviewed by: L. Hascoet and one anonymous referee

References

AdjoinableMPI: AdjoinableMPI wiki, https://trac.mcs.anl.gov/projects/AdjoinableMPI/wiki, last access: 19 October 2016.

ADOL-C: ADOL-C, http://www.coin-or.org/projects/ADOL-C.xml (last access: 19 October 2016), 2007.

Amestoy, P. R., Duff, I. S., Koster, J., and L'Excellent, J.-Y.: A Fully Asynchronous Multifrontal Solver Using Distributed Dynamic Scheduling, SIAM J. Matrix Anal. Appl., 23, 15–41, 2001.

Applegate, P. J., Kirchner, N., Stone, E. J., Keller, K., and Greve, R.: An assessment of key model parametric uncertainties in projections of Greenland Ice Sheet behavior, The Cryosphere, 6, 589–606, doi:10.5194/tc-6-589-2012, 2012.

Arthern, R. J. and Gudmundsson, G. H.: Initialization of ice-sheet forecasts viewed as an inverse Robin problem, J. Glaciol., 56, 527–533, 2010.

Aschwanden, A., Aðalgeirsdóttir, G., and Khroulev, C.: Hindcasting to measure ice sheet model sensitivity to initial states, The Cryosphere, 7, 1083–1093, doi:10.5194/tc-7-1083-2013, 2013.

Bindschadler, R., Nowicki, S., Abe-Ouchi, A., Aschwanden, A., Choi, H., Fastook, J., Granzow, G., Greve, R., Gutowski, G., Herzfeld, U., Jackson, C., Johnson, J., Khroulev, C., Levermann, A., Lipscomb, W., Martin, M., Morlighem, M., Parizek, B., Pollard, D., Price, S., Ren, D., Saito, F.and Sato, T., Seddik, H., Seroussi, H., Takahashi, K., Walker, R., and Wang, W.: Ice-Sheet Model Sensitivities to Environmental Forcing and Their Use in Projecting Future Sea-Level (The SeaRISE Project), J. Glaciol., 59, 195–224, doi:10.3189/2013JoG12J125, 2013.

Blatter, H.: Velocity And Stress-Fields In Grounded Glaciers: A Simple Algorithm For Including Deviatoric Stress Gradients, J. Glaciol., 41, 333–344, 1995.

Cornford, S., Martin, D., Graves, D., Ranken, D. F., Le Brocq, A. M., Gladstone, R., Payne, A., Ng, E., and Lipscomb, W.: Adaptive mesh, finite volume modeling of marine ice sheets, J. Comput. Phys., 232, 529–549, doi:10.1016/j.jcp.2012.08.037, 2013.

Galassi, M. E. A.: GNU Scientific Library Reference Manual, Network Theory Ltd, 3rd Edn., http://www.gnu.org/software/gsl/manual/gsl-ref.ps.gz (last access: 19 October 2016), 2009.

Giles, M.: Collected Matrix Derivative Results for Forward and Reverse Mode Algorithmic Differentiation, in: Advances in Automatic Differentiation, Springer Berling Heidelberg, Berling, Heidelberg, 35–44, 2008.

Glen, J.: The creep of polycrystalline ice, Proc. R. Soc. A, 228, 519–538, 1955.

Glen, J.: The flow law of ice: A discussion of the assumptions made in glacier theory, their experimental foundations and consequences, IASH Publ., 47, 171–183, 1958.

Goldberg, D. N. and Sergienko, O. V.: Data assimilation using a hybrid ice flow model, The Cryosphere, 5, 315–327, doi:10.5194/tc-5-315-2011, 2011.

Goldberg, D. N. and Heimbach, P.: Parameter and state estimation with a time-dependent adjoint marine ice sheet model, The Cryosphere, 7, 1659–1678, doi:10.5194/tc-7-1659-2013, 2013.

Goldberg, D. N., Heimbach, P., Joughin, I., and Smith, B.: Committed retreat of Smith, Pope, and Kohler Glaciers over the next 30 years inferred by transient model calibration, The Cryosphere, 9, 2429–2446, doi:10.5194/tc-9-2429-2015, 2015.

Griewank, A. and Walther, A.: Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, second Edn., 2008.

Griewank, A., Juedes, D., Mitev, H., Utke, J., Vogel, O., and Walther, A.: ADOL-C: A Package for the Automatic Differentiation of Algorithms Written in C/C++; this is the updated version of the paper published in ACM TOMS, vol. 22, June 1996, Algor, 755, 131–167, 1996.

Habermann, M., Maxwell, D., and Truffer, M.: Reconstruction of basal properties in ice sheets using iterative inverse methods, J. Glaciol., 58, 795–807, doi:10.3189/2012JoG11J168, 2012.

Habermann, M., Truffer, M., and Maxwell, D.: Changing basal conditions during the speed-up of Jakobshavn Isbræ, Greenland, The Cryosphere, 7, 1679–1692, doi:10.5194/tc-7-1679-2013, 2013.

Heimbach, P.: The MITgcm/ECCO adjoint modelling infrastructure, CLIVAR Exchanges, 44 (Volume 13, No. 1), 13–17, http://scholar.google.com/scholar?q=related:M8_kkL0Y1rUJ:scholar.google.com/&hl=en&num=20&as_sdt=0,5 (last access: 19 October 2016), 2008.

Heimbach, P. and Bugnion, V.: Greenland ice-sheet volume sensitivity to basal, surface and initial conditions derived from an adjoint model, Ann. Glaciol., 50, 67–80, 2009.

Khazendar, A., Rignot, E., and Larour, E.: Larsen B Ice Shelf rheology preceding its disintegration inferred by a control method, Geophys. Res. Lett., 34, 1–6, doi:10.1029/2007GL030980, 2007.

Larour, E., Rignot, E., Joughin, I., and Aubry, D.: Rheology of the Ronne Ice Shelf, Antarctica, inferred from satellite radar interferometry data using an inverse control method, Geophys. Res. Lett., 32, 1–4, doi:10.1029/2004GL021693, 2005.

Larour, E., Seroussi, H., Morlighem, M., and Rignot, E.: Continental scale, high order, high spatial resolution, ice sheet modeling using the Ice Sheet System Model (ISSM), J. Geophys. Res., 117, 1–20, doi:10.1029/2011JF002140, 2012.

Larour, E., Utke, J., Csatho, B., Schenk, A., Seroussi, H., Morlighem, M., Rignot, E., Schlegel, N., and Khazendar, A.: Inferred basal friction and surface mass balance of the Northeast Greenland Ice Stream using data assimilation of ICESat (Ice Cloud and land Elevation Satellite) surface altimetry and ISSM (Ice Sheet System Model), The Cryosphere, 8, 2335–2351, doi:10.5194/tc-8-2335-2014, 2014.

MacAyeal, D.: Large-scale ice flow over a viscous basal sediment: Theory and application to Ice Stream B, Antarctica, J. Geophys. Res., 94, 4071–4087, 1989.

MacAyeal, D.: A tutorial on the use of control methods in ice-sheet modeling, J. Glaciol., 39, 91–98, 1993.

Monnier, J.: DassFlow: Data Assimilation for Free Surface Flows, http://www.math.univ-toulouse.fr/DassFlow (last access: 19 October 2016), 2010.

Moon, T., Joughin, I., and Smith, B.: Seasonal to multiyear variability of glacier surface velocity, terminus position, and sea ice/ice mélange in northwest Greenland, J. Geophys. Res.-Earth, 120, 818–833, 2015.

Morlighem, M., Rignot, E., Seroussi, H., Larour, E., Ben Dhia, H., and Aubry, D.: Spatial patterns of basal drag inferred using control methods from a full-Stokes and simpler models for Pine Island Glacier, West Antarctica, Geophys. Res. Lett., 37, 1–6, doi:10.1029/2010GL043853, 2010.

Morlighem, M., Rignot, E., Mouginot, J., Seroussi, H., and Larour, E.: High-resolution ice thickness mapping in South Greenland, Ann. Glaciol., 55, 64–70, doi:10.3189/2014AoG67A088, 2014.

Nowicki, S., Bindschadler, R., Abe-Ouchi, A., Aschwanden, A., Bueler, E., Choi, H., Fastook, J., Granzow, G., Greve, R., Gutowski, G., Herzfeld, U., Jackson, C., Johnson, J., Khroulev, C., Larour, E., Levermann, A., Lipscomb, W., Martin, M., Morlighem, M., Parizek, B., Pollard, D., Price, S., Ren, D., Rignot, E., Saito, F., Sato, T., Seddik, H., Seroussi, H., Takahashi, K., Walker, R., and Wang, W.: Insights into spatial sensitivities of ice mass response to environmental change from the SeaRISE ice sheet modeling project I: Antarctica, J. Geophys. Res., 118, 1–23, doi:10.1002/jgrf.20081, 2013.

Pattyn, F.: Numerical modelling of a fast-flowing outlet glacier: experiments with different basal conditions, Ann. Glaciol., 23, 237–246, 1996.

Pawlowski, R. P., Phipps, E. T., and Salinger, A. G.: Automating embedded analysis, CoRR, abs/1205.0790, http://arxiv.org/abs/1205.0790 (last access: 19 October 2016), 2012.

Phipps, E. T., Bartlett, R. A., Gay, D. M., and Hoekstra, R. J.: Large-Scale Transient Sensitivity Analysis of a Radiation-Damaged Bipolar Junction Transistor via Automatic Differentiation, in: Advances in Automatic Differentiation, Springer Berlin Heidelberg, 351–362, 2008.

Price, S., Payne, A., Howat, I., and Smith, B.: Committed sea-level rise for the next century from Greenland ice sheet dynamics during the past decade, P. Natl. Acad. Sci. USA, 108, 8978–8983, 2011.

Rommelaere, V. and MacAyeal, D.: Large-scale rheology of the Ross Ice Shelf, Antarctica, computed by a control method, Ann. Glaciol., 24, 43–48, 1997.

Smart, J. F.: Jenkins: The Definitive Guide, O'Reilly Media, Inc., 2011.

Stokes, G.: On the theories of internal friction of fluids in motion, Trans. Cambridge Philos. Soc., 8, 287–305, 1845.

The Trilinos Project: Trilinos Home Page, http://trilinos.sandia.gov/ (last access: 19 October 2016), 2014.

Utke, J., Hascoët, L., Heimbach, P., Hill, C., Hovland, P., and Naumann, U.: Toward adjoinable MPI, 2009 IEEE International Symposium on Parallel & Distributed Processing, 1–8, 2009.