Author comment replying to the referee comment by L. Tarasov

We like to thank the reviewer for his comments on the manuscript and would hereby like to address the concerns he raised.

In Italics the comments, below our rebuttal

This study contrasts with my approach, briefly described in T & P 2006 that focused on coarsening the hydrological DEM resolution to the resolution of the ice sheet grid while preserving routing pathways. It would be worth a few sentences comparing the two approaches with respect to computational speed and accuracy given the different tradeoffs between the two approaches and the contextual accuracy of the ice margin.

We agree that a comparison of the two methods in terms of computational speed is of added value to the manuscript. Although we don't have the code from T & P 2006, we worked along the concepts of their algorithm during the start of our project, but quickly concluded that this approach was computationally more expensive. This is mainly because the drainage pointer approach must be applied to the whole region, meaning that, although it has a larger scope, it needs to operate on every grid cell. Our approach only treats the flooded grid cells of a designated drainage basin. For the case considered our code is a factor 5 faster. This will be described in a separate section in the manuscript

The last point needs to be underlined as the uncertainties in paleo ice sheet margins will always be much larger than 1 km (and I don't see 1 km grid resolution continental scale ice sheet models running glacial cycles anytime soon). Heck, there are few locations along the Laurentide ice sheet where we will confidently never know the ice margin location to even +/- 40 km resolution at any given time barring some new dating technique)

We agree that the uncertainty in ice margin reconstructions will likely never reach the 1 km resolution described in our manuscript, at least not everywhere. However, we believe that this does not detract from the added value of a 1 km lake reconstruction over a 40 km version. Most topographical features that would limit the water level of such a lake through draining, such as river valleys, have horizontal dimensions that are far smaller than 40 km, meaning that a 40 km analysis would overlook these features and thereby overestimate the water volume. Hence solving at 40 km introduces a systematic error, which is only partly related to the uncertainty in the location of the ice margin. A 1 km analysis strongly reduces this systematic error. We will add a few sentences to the "Introduction" section of the manuscript to clarify this point and quantify the difference in calculated water volume.

:: 29 Tarasov and Peltier, 2004).
 # inappropriate reference, should be Tarasov and Peltier, 2005 and 2006

We apologize for this erroneous reference and will correct this.

:: Lake Agassiz.... 6 It is therefore important to accurately model the extent and volume of the lake over time

 # Tarasov and Peltier, 2006 would I think be a relevant reference for this since they model Lake Agassiz (other North Am pro-glacial lake) evolution

We agree that this is a relevant reference and will add this to the manuscript.

::the largest of which is Lake Agassiz, along the southern margin of the ice-sheet. Lake Agassiz ... :: Doing this requires an accurate treatment of the large changes in the land/ocean-mask that occur where the ice-sheet covers most of the Canadian Arctic Archipelago and blocks the Hudson Strait. This changes the location where lake outflow reaches the sea over time

 # The above is geographically/geologically incorrect and has no relevance to Lake Agassiz. Neither the Canadian Arctic Archipelago (CAA) nor Hudson Strait ice were drainage blocks for Lake Agassiz. The possible northern drainage outlet for Lake Agassiz is the Mackenzie River delta which is outside of the CAA. Ice across Hudson Bay and Northern Ontario is what dammed the lake in the direction of Hudson Strait drainage according to consensus geological inferences (cf eg, Dyke, 2004). And the 8.2 ka (not 8.4) drainage was for proglacial Lake Ojibway not Agassiz.

We agree that the manuscript may be confusing here - indeed, we do not wish to suggest that any drainage events happened through the CAA. However, our manuscript does not concern any particular drainage event. It proposes a mathematical algorithm, which can be used as a tool to study such events. We have chosen a model-generated ice sheet configuration, which allows for a (perhaps unrealistically) large proglacial lake to form, because this creates the most computationally expensive setting, and therefore optimally illustrates the advantages of our approach - a configuration, which indeed may never have existed in reality. We will clarify this reasoning in the manuscript.

#Given that no one will be running 1km grid resolution ice sheet models for glacial cycle contexts (given "proglacial" in the title) in the foreseeable future, give the interpolation time to provide a complete time budget

We agree, and will provide these numbers in the manuscript.

: Supplement

I have not been able to test the code since the required netcdf files are not on this

server. But I have a few suggestions:

1) the ReadMe.txt should provide command line examples how to run the scripts (so that the reader doesn't have to dig right away into the code to see if there are any arguments that need passing).

We agree, and will add these examples to the ReadMe.txt file.

2) Verify that the code runs on octave. What is the point of using open source publishing to publish something that requires a close source app especially when an open source alternative is available?

We agree, and are currently working on this. We do not expect any trouble, since the codes only use very basic function calls, all of which (including the NetCDF package) are available in Octave. We will include functional Octave scripts in the supplementary material.

3) add the required net-cdf files for the sample scripts on the GMD page (as a separate supplement...)

The required NetCDF files are freely available online and can be found with their own separate DOI, which is mentioned in the "Code and data availability" section of the manuscript. However, we have indeed discovered that not all internet browsers and search engines handle DOI's equally. We will therefore add the URL for the NetCDF files.

Author comment replying to the referee comment posted by anonymous referee #2

We like to thank the reviewer for his comments on the manuscript and would hereby like to address the concerns raised.

In Italics the comments, below our rebuttal

While the work carried out looks to be robust and of value to a fairly specific application, I wonder whether the work has a broad enough reach to be published in a relatively high impact journal such as GMD. The first line of the paper suggests that the problem of determining lake depths is often encountered in many fields, but doesn't go on to give any examples beyond proglacial lakes. And do these other applications need to solve this problem over many iterations? Otherwise the computational time of minutes is not a massive issue, and the techniques presented have a fairly niche application, and could be presented in a paper specific to the application.

We agree that we could have expressed the relevance of the work a bit better than we did. We provided a couple of papers with applications in hydrology for which our work has relevance (Tarboton et al., 1991, Zhu et al., 2006, Goelzer et al., 2012 and references therein), where drainage direction maps need to be determined while accurately accounting for lakes within the maps. In addition, there are studies looking at changes in global hydrology either over past climate changes or future projections (e.g. Renssen and Knoop, 2000) where the boundary conditions of the problem may change over time. While we believe that hydrological studies benefit from our improved algorithm, we agree that the main application will probably still be the treatment of proglacial lakes within ice models. Having said we would like to stress that the dynamics of ice sheets is an emerging field of interest driven by the need to address the uncertainty in sea level projections and the appearance of more and more empirical data and paleoclimatological data. Meltwater pulses are for instance still poorly understood. What we hope to offer with manuscript is a generic tool for all ice sheet models, which they can use to test the importance of lakes in their specific area of interest, whether this is Laurentide, Eurasia, the past or present and future evolution of Greenland and Antarctica is up to the individual researchers. Given the importance for hydrology and dynamical ice sheet studies we believe this model deserves a separate paper rather than an appendix to a specific application. This thought is strengthened by the fact that we need a full paper to explain the merits of the approach.

We have improved the introduction to clarify this better and added the reference to the work by Renssen and Knoop.

The introduction does not give enough background to the task, such that I had to read it a number of times to understand exactly what the task at hand was.

We have improved the introduction in order to clarify the problem addressed in the paper..

Further there is not much detail on the default algorithm, the reader is left to go and read the paper by Zhu et al. (2006), such that it is a bit hard to follow independently.

We have improved section 2 with a figure and text to illustrate how the default algorithm works as a starting point for our own configuration.

There are also a lot of vague statements such as "A second issue is an efficient way to determine the water level" on page 3, line 22. Because you don't introduce your terms, such as "base level" and "water level" it is not always easy to follow the methods. Be more specific in these statements to help the reader understand what it is going on.

The term "base level" in the manuscript is indeed erroneous, this should read "local topography". The term "water level" means the elevation of the water surface with respect to sea level. We clarify this in the revised version of the manuscript.

Another example, on page 4, line 18 and Fig. 1, you talk about a true/false mask but don't define what this is. In Fig. 2 you talk about a land/ocean mask, but all I can see is DEM elevations? Is the deep blue actually a mask and not elevations? In this case, this needs to be in the legend.

The "true/false-mask" is the same one that is introduced on page 3, line 21. We will re emphasize this in the text. Fig. 2 does indeed show the DEM elevation with the land/ocean mask as a deep blue overlay. We will clarify this in the figure caption.

List of changes

Page 1, line 16: Added a line to the abstract to underline the range of possible uses for our algorithm
Page 1, line 19: Added several references to studies from different scientific fields that have made use of similar algorithms and that would therefore have benefited from our improvements.
Page 2, line 4: Changed the erroneous reference to Tarasov & Peltier, 2004 to the correct publication from 2006.
Page 2, line 12: Added an additional reference to Tarasov and Peltier, 2006.
Page 2, line 13: Changed the phrasing of the sentence describing the ice sheet configuration we used to more clearly explain what is happening.
Page 2, line 21: Changed the phrasing of the sentence describing how our ice sheet configuration was created to underline that it is not meant to represent reality at any specific point in time.
Page 2, line 23: Added a line stating that we compared the computational performance of our algorithm to the Zhu et al., 2006, version and to the drainage pointer approach used by Tarasov & Peltier, 2006.
Page 2, line 26: Added several lines to explain why the algorithm is applied to a 1 km resolution DEM.
Page 3, line 3: Replaced the term "base level" by "local topography" and defined the term "water level".
Page 3, line 16: Added a line referring to the new Figure 1, which illustrates how the default flood-fill algorithm works.
Page 4, line 1: Clarified the term "land/ocean mask" and how it is created by the algorithm.
Page 5, line 7: Fixed a minor grammatical error.
Page 9, line 20: Added a paragraph describing the drainage pointer approach used by Tarasov & Peltier, 2006. Added lines that describe the experiment comparing their approach to ours in terms of computational performance.
Page 10, line 5: Clarified the term "default algorithm".
Page 10, line 12: Added the required computation time for interpolating the input fields required by the algorithm.
Page 10, line 27: Added a direct link to the data repository where the NetCDF files required to run the demo codes from the supplementary material can be found.
Page 11: Added several references that are referred to on page 1, line 19.
Page 13: Added an extra figure illustrating how the default flood-fill algorithm works, with a description in the caption. References to all figures have been updated accordingly.
Page 14, line 7: Clarified how the land/ocean-mask is depicted in Figure 3.

A computationally efficient depression-filling algorithm for digital elevation models applied to proglacial lake drainage

Constantijn J. Berends[1], Roderik S. W. van de Wal[1]

[1]Institute for Marine and Atmospheric research Utrecht, Utrecht University, Utrecht, 3584 CC, The Netherlands

Correspondence to: Constantijn J. Berends (c.j.berends@uu.nl)

Abstract. We present and evaluate several optimizations to a standard flood-fill algorithm in terms of computational efficiency. As an example, we determine the land/ocean-mask for a 1 km resolution digital elevation model (DEM) of North America and Greenland, a geographical area of roughly 7000 by 5000 km (roughly 35 million elements), about half of which is covered by ocean. Determining the land/ocean-mask with our improved flood-fill algorithm reduces computation time by 90 % relative to using a standard stack-based flood-fill algorithm. In another experiment, we use the bedrock elevation, ice thickness and geoid perturbation fields from the output of a coupled ice-sheet - sea-level equation model at 30,000 years before present and determine the extent of Lake Agassiz, using both the standard and improved versions of the flood-fill algorithm. We show that several optimizations to the flood-fill algorithm used for filling a depression up to a water level, that is not defined at forehand, decrease the computation time by up to 99 %. The resulting reduction in computation time allows determination of the extent and volume of depressions in a DEM over large geographical grids or repeatedly over long periods of time, where computation time might otherwise be a limiting factor. The algorithm can be used for all glaciological and hydrological models, which need to trace the evolution over time of lakes or drainage basins in general.

1 Introduction

The problem of determining drainage routing maps is often encountered in scientific fields such as hydrology (Renssen and Knoop, 2000), (palaeo-) climatology (Goelzer et al., 2012) and glaciology (Marshall et al, 1999, Tarasov and Peltier, 2006). Several studies have underlined the importance of accounting for depressions when creating such drainage maps from DEM's (Zhu et al., 2006, Arnold, 2010).

An important example is the routing of melt-water and the drainage of proglacial lakes that formed at the margins of the Laurentide ice-sheet. The retreat of the Laurentide ice-sheet during the last deglaciation and the corresponding release of large fluxes of fresh water into the Arctic Sea and the northern Atlantic Ocean have been linked to climatic events through disrupting the Atlantic Meridional Overturning Circulation (Barber et al., 1999, Clark et al., 2001, Li et al., 2012, Tarasov and Peltier, 2005, Teller et al., 2002). Melt-water from the remnants of Northern Hemisphere ice-sheets has also shown to have influenced the climate system during the previous interglacial (Stone et al., 2016).

Many studies have focused on reconstructing the direction and magnitude of the freshwater flux from geological data (Broecker et al., 1989, Hillaire-Marcel et al., 2007, LaJeunesse and St.-Onge, 2008, Törnqvist et al., 2004). More recently

these freshwater fluxes have been estimated by stand-alone ice-sheet models (Goelzer et al., 2012, Marshall et al., 1999, Tarasov and Peltier, 2006).

Due to the influence of the local topography, part of the melt-water from the retreating ice-sheet was not directly released into the ocean system but temporarily stored in proglacial lakes, the largest of which is Lake Agassiz, along the southern margin of the ice-sheet. Lake Agassiz is estimated to have covered 7.1×10^5 km^2 and contained approximately 1.4×10^{14} m^3 of water or 0.4 m global mean sea-level equivalent around 8.4 kyr BP, immediately prior to its catastrophic drainage (Kendall et al., 2008). The size of the flood is however poorly constrained and higher numbers have also been published (e.g. Hijma and Cohen, 2010). It is therefore important to accurately model the extent and volume of the lake over time, since the presence of the lake affects both the timing and location of the melt-water release into the ocean system, as well as the local climate (Krinner et al. 2004, Tarasov and Peltier, 2006). Doing this requires an accurate treatment of the large changes in the land/ocean-mask that occur, for example, where the ice-sheet at times covers most of the Canadian Arctic Archipelago or blocks the Hudson Strait. This changes the location where spill over from the lake and drainage events reach the sea over time. The flood-fill algorithm can be used to determine the land/ocean-mask and account for this effect.

When applying commonly used algorithms (Arnold, 2010, Doll and Lehner, 2002, Tarboton et al., 1991, Zhu et al., 2006) to a problem involving such large geographical grids and long time-scales, computation time can become a limiting factor, particularly when the geometry is changing over time and the procedure has to be repeated over many time steps.

In this study, we describe and evaluate several improvements to a standard algorithm for filling depressions in a DEM in order to improve the computational efficiency. The improved algorithm is applied to a 1 km resolution DEM, including an ice thickness distribution of North America 30,000 years ago generated by an ice sheet model in such a way as to create boundary conditions allowing for the formation of a very large lake. We determine both the land/ocean-mask and the size and extent of the proglacial lake for this glacial configuration and compare the required computation time with the default flood-fill algorithm as presented by Zhu et al. (2006) and with the drainage pointer approach presented by Tarasov and Peltier (2006).

The algorithm is applied at a horizontal resolution of 1 km, which is very high for such models. However, a lower resolution would overlook topographical features, such as small river valleys, that would limit the water level of the lake through drainage. This would lead to a systematic overestimation of the total water volume unrelated to the uncertainty of the modelled location of the ice sheet margin, which is usually not as high as 1 km.

A comparison between the calculated water volume of a lake at 1 km resolution versus the calculated volume of the same lake at 40 km resolution showed that the difference can be as large as 20 %.

2 Methodology

2.1 Default algorithm

The problem of filling a depression up to a pre-defined level can be likened to filling a hole in a true/false-mask (whether or not the ~~base level~~local topography is below the~~an~~ a priori chosen water level of the lake or ocean). There are several algorithms for solving this problem, generally known as "flood-fill" algorithms (Zhu et al., 2006). They are commonly known for their use in the "bucket" tool of several paint programs. A thorough description of the stack-based flood-fill algorithm is given by Zhu et al. (2006). Here we improve on this work, taking code written after their approach as a starting point. The algorithm starts with a "seed", defined as a designated element from where neighbouring elements are flooded. In many hydrological applications, the seed will be a local minimum in a DEM. Beside the seed element, the algorithm builds a "stack", which is an array listing the indices of the map elements. During each iteration of the algorithm, all stack elements are checked. If the elevation of a stack element is below the water level its corresponding map element is filled and it is removed from the stack and all its neighbours are added to the stack. If the element does not lie below the water level it is removed from the stack without any further action. The iterations are continued until the stack is empty and the elements below the water level are identified.

A pseudo-code example of this algorithm is illustrated below. An illustration of the algorithm being applied to a simple true/false mask is shown in Figure 1.

```
stack = seed

WHILE (stack is not empty)
 FOR (all elements in stack)
  IF (stack element lies below water level)
   remove element from stack
   FOR (8 neighbours of stack element)
    IF (neighbour is not filled and is not in stack)
     add neighbour to stack
    END IF
   END FOR
  END IF
 END FOR
END WHILE
```

An example of a Matlab implementation of this algorithm is provided in the supplementary material, being script "fill_1km.m".

The number of operations required for filling a hole with this algorithm is approximately proportional to the number of elements of the depression. This means that application to a very large area with a high resolution can result in a long computation time, since doubling the resolution in the horizontal plane will quadruple the number of operations.

A second issue is an efficient way to determine the water level. When using the algorithm to determine which elements will be filled by the ocean~~When creating the land/ocean mask~~, the water level is known and the true/false-mask that must be filled by the algorithm is created by checking for every single element whether its elevation is above or below ~~the water~~sea level. In the case of a lake, the water level can either be determined by the water mass balance, with inflow and evaporation in equilibrium. Alternatively, in the case of inflow that is not balanced by evaporation, the surrounding topography will limit the water level. In either case, the general solution to determining the water level is to start at the bottom of the depression and incrementally increase the water depth, either until the calculated evaporation becomes large enough to offset the inflow, or until the lake overflows.

A pseudo-code example of this algorithm is illustrated below.

```
water level = 0

WHILE (threshold is not reached)
 increase water level
 perform flood-fill
END WHILE
```

The threshold condition can be the overflowing of the lake into the sea, the total evaporation over the lake balancing total water influx, or any other logical condition.

Depending on the vertical resolution of the DEM and the accuracy required to determine the lake depth, such a calculation requires dozens to hundreds of iterations. This is not always a restriction for practical applications where one is interested in the water runoff pathways and drainage basins for a given DEM. However, when performing an ensemble of simulations, or a simulation where the topography changes over time, reducing the computation time can be crucial for the performance and feasibility of the application and optimizations are required. In the next paragraph we present several optimizations to the standard flood-fill algorithm, which reduce computation time considerably for map with a large area or a high resolution.

2.2 Optimizations

2.2.1 Low-resolution block inspection

When filling a depression to a pre-determined water level, we strongly reduce the number of operations by creating a lower resolution “maximum topography” map. For example, any element of a 4 km maximum topography map will contain the highest value of the corresponding 4 x 4 block of 1 km elements. Consequently, we apply the flood-fill algorithm to this lower resolution map. If the highest element of a 4 km x 4 km square lies below the water level, all sixteen 1 km elements must do, implying all of them can be flooded at once. This step yields a first coarse filling scheme without testing each individual element at the final higher resolution.

When the algorithm is finished, some elements may not yet be flooded where they should be, but the large majority of them will be filled already if the distribution is not too scattered. To identify those elements that still need to be flooded, we take this intermediate map and stack from the 4 km fill and use these as a starting point for a 1 km fill. Applied to the 1 km map, the algorithm will only have to fill in the "fringes" of the depression - a much smaller area than has already been processed by the low-resolution fill. Figure 2 illustrates this.

In Fig.1, the 1 km true/false-mask is shown as a black overlay in all panels. The 4 km stack (light green) is initiated with a seed in the lower left corner of the map, panel A. The 4 km algorithm expands from this seed until no more elements can be added. The final map (blue) and stack (light green) are shown in panel B. These are then converted to their 1 km equivalent, shown in panel C. As can be seen, the conversion conserves the filled elements. The 1 km algorithm then continues from this map and stack until no more elements can be added. The resulting map and stack coincides with the 1 km true/false-mask in panel D.

The decrease in computation time resulting from using the high-resolution algorithm is larger than the additional computation time required to create the low-resolution map and to convert the low-resolution stack to its high-resolution equivalent. This is shown for an example in Sect. 3.1.

A further increase in efficiency can be achieved by creating an additional medium-resolution map in between. For example, we can start with a 10 km fill, and then convert the resulting map and stack to 5 km resolution. Use the intermediate result for a 5 km fill algorithm, run this until completion, convert the resulting map and stack to their 1 km equivalents, and finalize with the 1 km fill. Note that this will only work when the different resolutions are integer multiples of each other. Constructing rules for when this condition is not met is not evident, and will increase computation time, and is beyond the scope of this study.

A Matlab implementation of the fill algorithm is provided in the supplementary material, being script "Demo_FillSea.m".

2.2.2 Shoreline memory

In the case of a depression that must be filled up to overflow conditions, one small change to the flood-fill algorithm is straightforward. Any element that is filled at a certain water depth will also be filled for any higher water depth, meaning the lake's shoreline will only expand outward when the water depth is increased. Hence, in the improved flood-fill algorithm, a stack element that is found to lie above the water level is not removed from the stack. Instead it is flagged and not inspected again in any further loops. This means that when the algorithm is finished, the stack contains all those and only those elements, which directly border filled elements but which are not filled, being the shoreline.

When the water depth is increased during the filling of a depression, the final stack from the previous loop is taken as a starting point. This means that all elements of the final, deepest lake have only been inspected once during the iteration procedure. For a deep lake, which requires many depth increments to be filled, the decrease in the required computation time can be large in this way, as will be shown in Sect. 3.3.

2.2.2 Low-resolution lake depth estimation

When trying to fill a depression up to the level of overflowing, i.e. with no pre-determined water level, it is generally not possible to use the low-resolution block inspection algorithm presented in Sect. 2.2.1. This is because the algorithm needs to check if overflow is reached for every single depth increment, and must therefore perform a 1 km fill at every depth increment. As converting a high-resolution stack and map back to the lower resolution is not possible, we cannot use the final high-resolution shoreline to start a new low-resolution fill.

~~This issue is solved by performing a fill on a low-resolution average topography map~~Performing a fill on a low-resolution average topography map solves this issue. Although not mathematically necessary, the topography of a geographical area is usually smooth enough such that a low-resolution estimate will yield a water depth relatively close to the "true" water depth that would be calculated with a high-resolution fill. This means that, instead of starting at zero depth, we can initiate the algorithm with a depth slightly below the depth yielded by the low-resolution estimate. The lake at this depth will be close to its maximum extent, so we can use the block inspection method to efficiently fill the majority of the lake's central elements at low resolution, thus reducing the computation time. The lake's fringes are filled with the high-resolution algorithm and the depth is increased incrementally until overflow is reached. Section 3.4 describes an experiment where this method is implemented. A pseudo-code example of this algorithm is illustrated below.

```
water level = 0

WHILE (threshold is not reached)
 increase water level
 perform flood-fill on 40 km average topography field
END WHILE

decrease water level
perform flood-fill on 40 km maximum topography field
convert 40 km map and stack to 1 km equivalent

WHILE (threshold is not reached)
 increase water level
 perform flood-fill on 1 km topography field
END WHILE
```

A Matlab implementation of this algorithm is provided in the supplementary material, being script "Demo_FillLake.m".

3 Results

All experiments are performed in Matlab R2014b on a 2013 iMac with a 3.2GHz Intel i5 processor and 8GB 1600MHz DDR3 Memory. Note that Matlab is an interpreted language and that the performance of the algorithm in a compiled language such as Fortran or C will generally be much faster. However, the relative improvements in performance of the optimized algorithms should be preserved.

3.1 Low-resolution block inspection

As a first example, we use different versions of the flood-fill algorithm to determine the land/ocean-mask for a DEM of North America and Greenland (Amante and Eakins, 2009), using an oblique stereographic projection at 1 km resolution. The region covers an area of roughly 7000 by 5000 km, resulting in approximately 17 million ocean elements to be filled. This extremely large size is useful for testing the efficiency of different set-ups and is needed in order to model the pressure exerted by lakes in gravitationally consistent calculations during the evolution of ice-sheets in North America.

Creating the land/ocean mask depicted in Fig. 3 with the standard version of the flood-fill algorithm at a 1 km resolution takes approximately 82 seconds. This serves as benchmark to which all subsequent experiments will be compared.

We performed two series of experiments with block inspection at different resolutions. The first series considers a sequence of 40 km, 8 km, 4 km, 2 km and 1 km resolutions. The second series considers a sequence of 40 km, 20 km, 10 km, 5 km and 1 km resolutions. The results in terms of computational efficiency are presented in Table 1 and Table 2, respectively.

Both the conversion and filling parts of the Tables can be read as "coming from [row] resolution, going to [column] resolution". For example (Table 1; highlighted in yellow), converting a 40 km map and stack to their 1 km equivalent takes 0.20 seconds. A 1 km fill starting with that map and stack will then take 27.54 seconds. The diagonal elements in the filling parts of the tables indicate the computation time of a fill at that resolution starting from a single seed element.

An overview of several possible resolution schemes is given in Table 3. It shows that the most efficient scheme (40 km - 5 km - 1 km) is almost an order of magnitude faster than the default algorithm at 1 km resolution.

3.2 Shape dependence

The total gain in efficiency from the improvements to the algorithm will of course depend on the details of the problem it is applied to. In the optimized algorithm, the number of elements that can be filled at the low resolution roughly scales with the area of the depression, whereas the number of elements that need to be filled at the high resolution roughly scales with the depression's circumference. The gain in computational efficiency therefore depends largely on the shape of the depression, becoming larger when the depression becomes more circular.

To illustrate this, the improved algorithm with a 40 km - 5 km - 1 km resolution scheme was applied to several versions of the same topography field, passed through increasingly strong high-pass filters with a cut-off wavelength of 30km, so that

the remaining topographical features that determine the shape of the depression are all too small to be visible on the low-resolution map in Fig. 4.

The results of the experiment are given in Table 4. The area of the ocean basin and therefore the number of elements that require filling does not change much. However, the compactness factor C of the ocean basin, which is defined as:

$$C = \frac{A}{4\pi\emptyset^2}$$

with ∅ the circumference and A the area of the basin, decreases sharply when the small-scale topographical features become more prominent.

This illustrates that the gain in computational efficiency from the block inspection depends strongly on the shape of the topography in which a depression must be filled. Natural topographical features generally have a vertical scale proportional to their horizontal scale, which leads to the relatively circular shapes of most natural lakes. This means that, for most natural topographies, the block inspection algorithm will substantially decrease the computation time required for filling depressions.

3.3 Shoreline memory

Similar to the previous experiment, we consider the lake formation in the North American region 30,000 years ago as a second example. At this time, large parts of the North American continent were covered by the Laurentide ice-sheet. The depression left in the bedrock by the weight of the ice, combined with the mass of ice damming off the Hudson Strait lead to the formation of a massive proglacial lake over the area of what is now known as the Hudson Bay. The situation is shown in Fig. 5.

We apply the flood-fill algorithm to the same 1 km resolution topographic map, combined with an ice thickness distribution, a bedrock deformation field and a geoid perturbation field (all generated by the ANICE-SELEN model - de Boer et al, 2014) to determine the size of the lake. These data fields have a 40 km resolution and are used to perturb the 1 km resolution DEM, such that the small topographic features are preserved. The 40 km resolution perturbation fields are mapped onto the 1 km resolution data fields using bilinear interpolation. The lake is found to cover an area of 3.5×10^6 km^2, with a volume of 6.0×10^{14} m^3 or about 1.7 meters eustatic sea-level equivalent of water.

As a benchmark, we apply the standard flood-fill algorithm without any optimizations, using a 5 m depth increment. This means that the algorithm fills the depression up to a certain water level, increases the depth by 5 m and fills the depression up to the new water level from the same seed, until the lake overflows into the sea. This run costs 351 seconds (almost six minutes) to complete. This implies that for a 120,000 year period which is resolved every 5 years, which is a time step typically used to resolve the last glacial cycle in an ice-sheet model, computational time is excessive relative to the approximately 50 hours such a simulation would otherwise require.

Performing the same test with a depth increment of 10 m takes 179 seconds. This is far less because for each depth interval the complete lake has to be filled, so the number of fills increases more or less linearly with the requested accuracy. A 5 m depth interval leads to an accuracy of about 3 % in the lake's volume, which is considered to be sufficiently accurate.

By implementing the "shoreline memory" improvement described in 2.2.2 to the algorithm we improve the performance in terms of computation time. As explained earlier, the improved shoreline memory implies that every element will only be checked once, yielding a drastic reduction in computation time, which is particular relevant for large lakes. Filling the same lake at 1 km resolution only takes 23 seconds in the optimized case, a reduction in computation time of about 90 % with respect to the benchmark.

3.4 Low-resolution lake depth estimation

As a final improvement, we implement the depth estimation method. This method starts by filling the same lake on the 40 km resolution topography, in order to get a depth estimate. The resulting lake is depicted in Fig. 5. It has a volume of 6.6×10^{14} m^3, overestimating the "true" 1 km volume by about 10 %. The 40 km estimate covers several river valleys visible in Fig. 6c, which should drain the lake. Since these valleys are only a few kilometres wide they do not appear on the smoothed 40 km resolution topography map. This causes the overestimation of the water level and the lake's extent and volume - hence a high resolution is needed. The resulting depth is reduced by an arbitrary amount of 20 % and a fill to this new depth is performed on the 40 km maximum topography field. The resulting lake, which will serve as a starting point for the final 1 km fill, is depicted in Fig. 6b.

The water level is then incrementally increased, while using the 1 km fill algorithm, until the lake overflows into the sea. The complete process takes 4.4 seconds to complete. This is about 5 times faster than the 1 km fill with shoreline memory and about 80 times faster than the default implementation, a reduction in computation time of about 99 % with respect to the benchmark experiment.

Note that the 20 % reduction in depth from the 40 km estimate is chosen somewhat arbitrarily. Depending on the smoothness of the local topography, it is possible that the 40 km estimate will be much more accurate. In such cases, it is justifiable to reduce the depth less, and hence improve efficiency even more. For this reason, it is advisable to check if the depth reduction from the 40 km estimate is enough to prevent overflow. If not, the depth needs to be increased even further before starting the 40 km maximum topography fill.

3.4 Comparison to the drainage-pointer method

As a benchmark of our method, we compare the computational performance of our optimized algorithm to that of the commonly used drainage-pointer approach (Tarasov and Peltier, 2006). This method assigns to every map element a drainage pointer, indicating which one of its immediate neighbours receives run-off from that element. By following the run-

off from an element until it reaches the sea, elements can be grouped together by the location where run-off reaches the sea, thus creating drainage basins.

We found that a Matlab implementation of the first step of this approach, where drainage pointers are assigned to all high resolution elements, required about five times more computation time than our optimized algorithm when used for determining the land/ocean mask as in section 3.1. The reason for this is that the drainage-pointer approach needs to assign these pointers to every single high resolution map element in order to work, whereas our method only needs to treat those elements that are beneath the water level and then only those near the shoreline at high resolution, thus limiting the number of operations.

4 Conclusions

We have presented and evaluated several optimizations to a standard flood-fill algorithm. When determining the land/ocean-mask over a large grid, the optimized algorithm is up to 9 times faster than the default algorithm. When determining the extent and depth of Lake Agassiz from a given DEM and ice thickness map, the optimized algorithm is even up to 80 times faster than the default algorithm as proposed by Zhu et al. (2006). The gain in computational efficiency depends on the smoothness of the topography, with the largest reduction in computation time achieved when most of the volume of the depression is contributed by topographical features with a horizontal scale larger than the lowest resolution of the block inspection method, as has been explained in section 2.2.2.

In the example given in this study, the ice thickness, bedrock deformation and geoid anomaly all initially have a 40 km resolution. In order to do a 1 km lake fill, these fields were interpolated onto a 1 km grid, which is computationally expensive (about 11 seconds in our Matlab implementation). If all input fields are already initially available at high resolution, they only need to be downscaled to a low resolution for the block inspection step, which takes considerably less time. For this reason, the computation time for this interpolation step is not included in the results.

The algorithm presented takes into account the bedrock deformation and geoid perturbation caused by the mass changes of an ice-sheet, if they have been calculated. These effects both influence the volume of a lake by deepening the basin because of the vertical water pressure on the landscape and by raising the water surface.

Any study involving either a very large grid or a large number of repeated simulations will greatly benefit from these optimizations. Examples include the modelling of large proglacial lakes, the dynamical modelling of run-off over an evolving ice-sheet, changes of water routing by tectonic activity and changes in a large water basin due to sea-level changes such as in the Mediterranean.

Code and data availability

Several Matlab scripts containing different versions of the flood-fill algorithm, as referenced in this manuscript, are provided as supplementary material. Two large NetCDF files containing data files required to run these scripts are available online at doi:10.5194/gmd-2016-85-supplement, or directly at https://zenodo.org/record/49614.

Acknowledgements

The Ministry of Education, Culture and Science (OCW), in the Netherlands, provided financial support for this study via the program of the Netherlands Earth System Science Centre (NESSC). Heiko Goelzer, Lennert Stap and Sarah Bradley commented on an earlier draft of this paper.

References

Amante, C. and Eakins, B. W.: ETOPO1 Arc-Minute Global Relief Model: Procedures, Data Sources and Analysis, National Oceanic and Atmospheric Administration Technical Memorandum NESDIS NGDC-24, doi: 10.7289/V5C8276M, 2009.

Arnold, N.: A new approach for dealing with depressions in digital elevation models when calculating flow accumulation values, Prog. Phys. Geog., 34, 781-809, doi: 10.1177/0309133310384542, 2010.

Barber, D. C., Dyke, A., Hillaire-Marcel, C., Jennings, A. E., Andrews, J. T., Kerwin, M. W., Bilodeau, G., McNeely, R., Southon, J., Morehead, M. D. and Gagnon, J.-M.: Forcing of the cold event of 8,200 years ago by catastrophic drainage of Laurentide lakes, Nature, 34, 781-809, doi: 10.1038/22504, 1999.

Broecker, W. S., Kenett, J. P., Flower, B. P., Teller, J., Trumbore, S., Bonani, G. and Wolfli, W.: Routing of meltwater from the Laurentide Ice Sheet during the Younger Dryas cold episode, Nature, 341, 318-321, doi: 10.1038/341318a0, 1989.

Clark, P. U., Marshall, S. J., Clarke, G., Hostetler, S. W., Licciardi, J. M. and Teller, J.: Freshwater Forcing of Abrupt Climate Change During the Last Glaciation, Science, 293, 283-287, doi: 10.1038/341318a0, 2001.

De Boer, B., Stocchi, P. and van de Wal, R. S. W.: A fully coupled 3-D ice-sheet-sea-level model: algorithm and applications, Geosci. Model Dev., 7, 2141-2156, doi: 10.5194/gmd-7-2141-2014, 2014.

Döll, P. and Lehner, B.: Validation of a new global 30-min drainage direction map, J. Hydrol., 258, 214-231, doi: doi:10.1016/S0022-1694(01)00565-0, 2002.

Goelzer, H., Janssens, I., Nemec, J. and Huybrechts, P.: A dynamic continental run-off routing model applied to the last Northern Hemisphere deglaciation, Geosci. Model Dev., 5, 599-609, doi:10.5194/gmd-5-599-2012, 2012.

Hijma, M. P. and Cohen, K. M.: Timing and magnitude of the sea-level jump preluding the 8200 yr event, Geology, 38, 275-278, doi: 10.1130/G30439.1, 2010.

Hillaire-Marcel, C., deVernal, A. and Piper, D. J.: Lake Agassiz final drainage event in the northwest North Atlantic, Geophys. Res. Lett., 34, 1-5, doi: 10.1029/2007GL030396, 2007.

Kendall, R., Mitrovica, J. X., Milne, G. A., Törnqvist, T. E., and Li, Y.-X.: The sea-level fingerprint of the 8.2 ka climate event, Geology, 36, 423-426, doi: 10.1130/G24550A.1, 2008.

Krinner, G., Mangerud, J., Jakobsson, M., Crucifix, M., Ritz, C. and Svendsen, J. I.: Enhanced ice sheet growth in Eurasia owing to adjacent ice-dammed lakes, Nature, 427, 429-432, doi: 10.1038/nature02233, 2004.

Lajeunesse, P. and St-Onge, G.: The subglacial origin of the Lake Agassiz-Ojibway final outburst flood, Nat. Geosci., 1, 184-188, doi: 10.1038/ngeo130, 2008.

Li, Y.-X., Törnqvist, T. E., Nevitt, J. M. and Kohl, B.: Synchronizing a sea-level jump, final Lake Agassiz drainage and abrupt cooling 8200 years ago, Earth Planet. Sc. Lett., 315, 41-50, doi: 10.1016/j.epsl.2011.05.034, 2012.

Marshall, S. J. and Clarke, G.: Modelling North American Freshwater Runoff through the Last Glacial Cycle, Quaternary Res., 52, 300-315, doi: 10.1006/qres.1999.2079, 1999.

Stone, E. J., Capron, E., Lunt, D. J., Payne, A. J., Singarayer, J. S., Valdes, P. J. and Wolff, E. W.: Impact of melt water on high latitude early Last Interglacial climate, Clim. Past., 11, doi:10.5194/cp-2016-11, 2016.

Tarasov, L. and Peltier, W. R.: A geophysically constrained large ensemble analysis of the deglacial history of the North American ice-sheet complex, Quaternary Sci. Rev., 23, 359-388, doi:10.1016/j.quascirev.2003.08.004, 2004.

Tarasov, L. and Peltier, W.R.: A calibrated deglacial chronology for the North American continent: evidence of an Arctic trigger for the Younger Dryas, Quaternary Sci. Rev., 25, 659-688, doi:10.1016/j.quascirev.2005.12.006, 2006.

Tarboton, D. G., Bras, R. L. and Rodriguez-Iturbe, I.: On the extraction of channel networks from digital elevation data, Hydrol. Process., 5, 81-100, 1991.

Teller, J., Leverington, D. and Mann, J.D.: Freshwater outbursts to the oceans from glacial Lake Agassiz and their role in climate change during the last deglaciation, Quaternary Sci. Rev., 21, 879-887, doi:10.1016/S0277-3791(01)00145-7, 2002.

Törnqvist, T. E., Bick, S. J., Gonzalez, J. L., van der Borg, K. and de Jong, A. F.: Tracking the sea-level signature of the 8.2 ka cooling event: New constraints from the Mississippi Delta, Geophys. Res. Lett., 31, doi: 10.1029/2004GL021429, 2004.

Zhu, Q., Tian, Y. and Zhao, J.: An efficient depression processing algorithm for hydrologic analysis, Computers and Geosciences, 32, 615-623, doi:10.1016/j.cageo.2005.09.001, 2006.

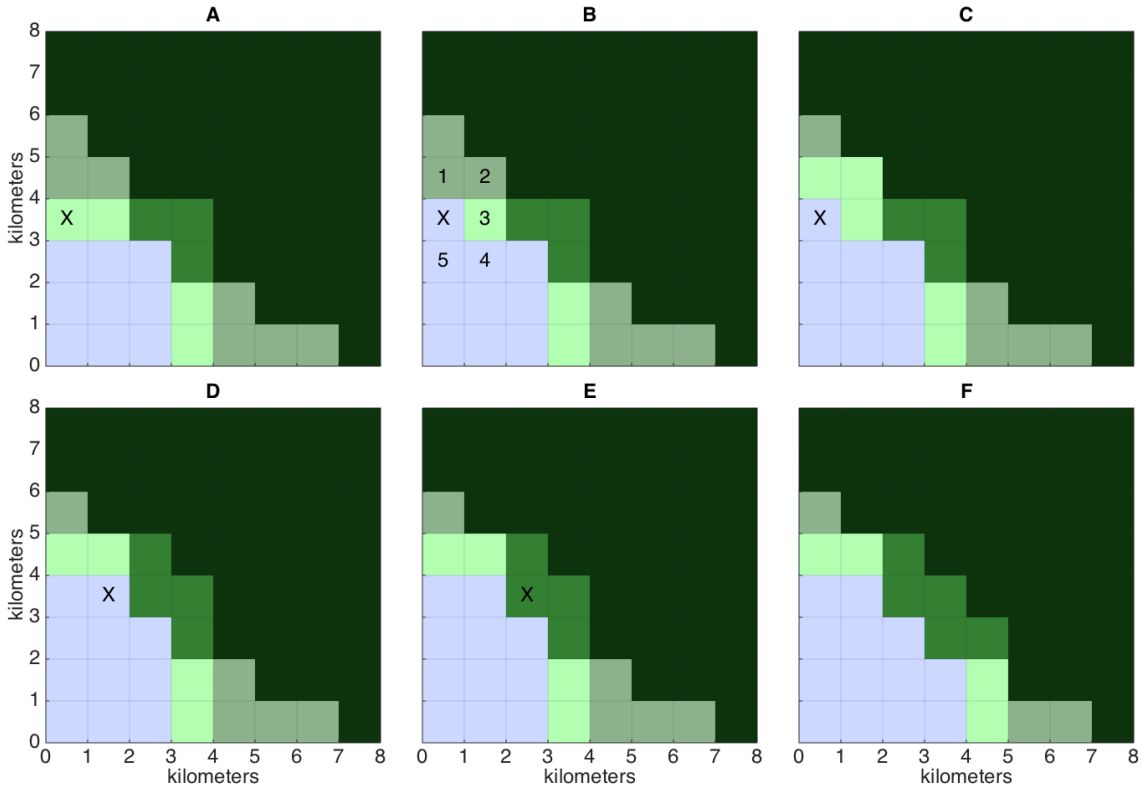

Figure 1: An illustration of the default flood-fill algorithm. Black overlay: true/false-mask: dark green: unfilled; light green: stack; blue: filled. A: the algorithm starts at the first element in the stack (marked with X). B: the element is found to meet the fill criterion and is filled. Its immediate neighbours (marked 1 to 5) are now inspected. C: Neighbours 1 and 2 were not yet in the stack and so they were appended to it. Neighbour 3 was already in the stack and neighbours 4 and 5 were already filled. D: The algorithm now moves on the next element in the stack (marked with X), which is filled and its top-right neighbour is added to the stack. E: The next element in the stack is found not to meet the fill criterion and so nothing is done. F: The result after inspecting the whole stack.

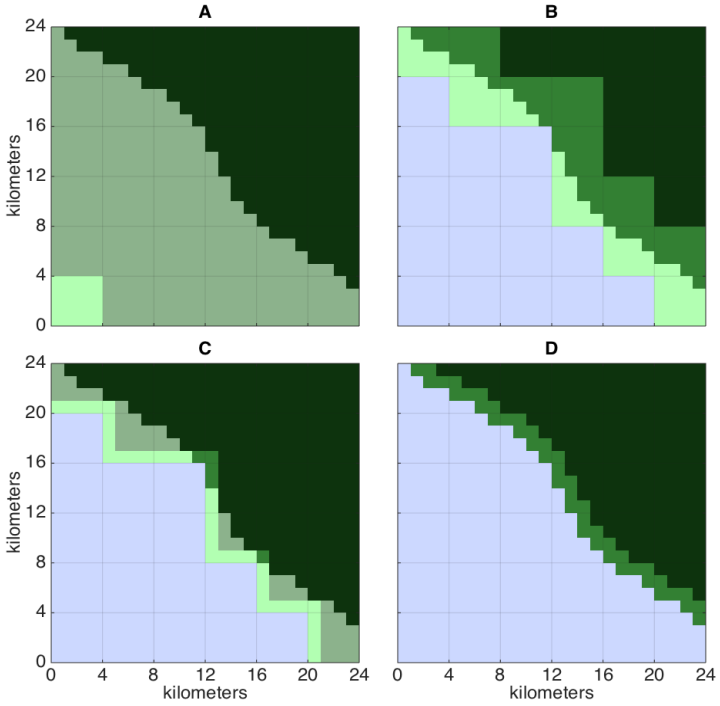

Figure 2: An illustration of the low-resolution block inspection improvement upon the default flood-fill algorithm. Black overlay: 1 km true/false-mask: dark green: unfilled; light green: stack; blue: filled. A: the 4 km fill is given a seed in the southwest corner of the map. B: the 4 km fill is completed. C: The 1 km map and edge created from the intermediate 4 km output. D: the result of the 1 km fill.

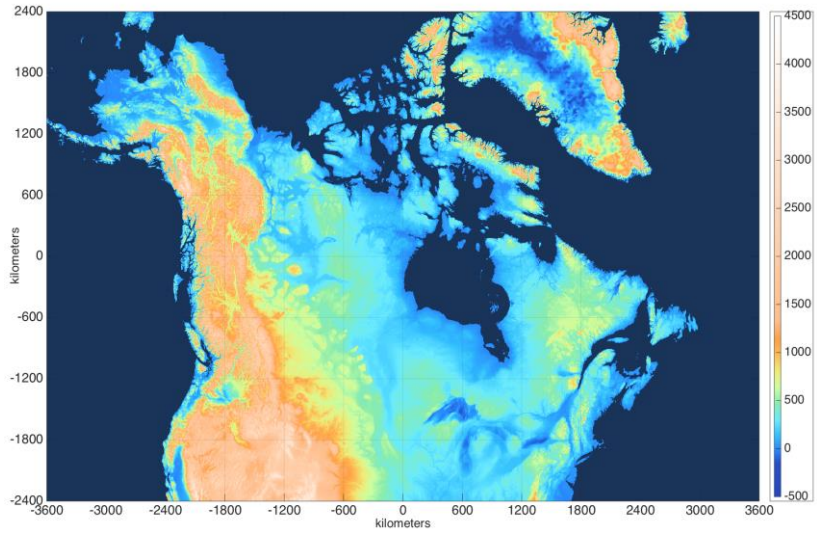

Figure 3: The present-day land/ocean-mask (uniform deep blue) and the bedrock topography at 1 km resolution.

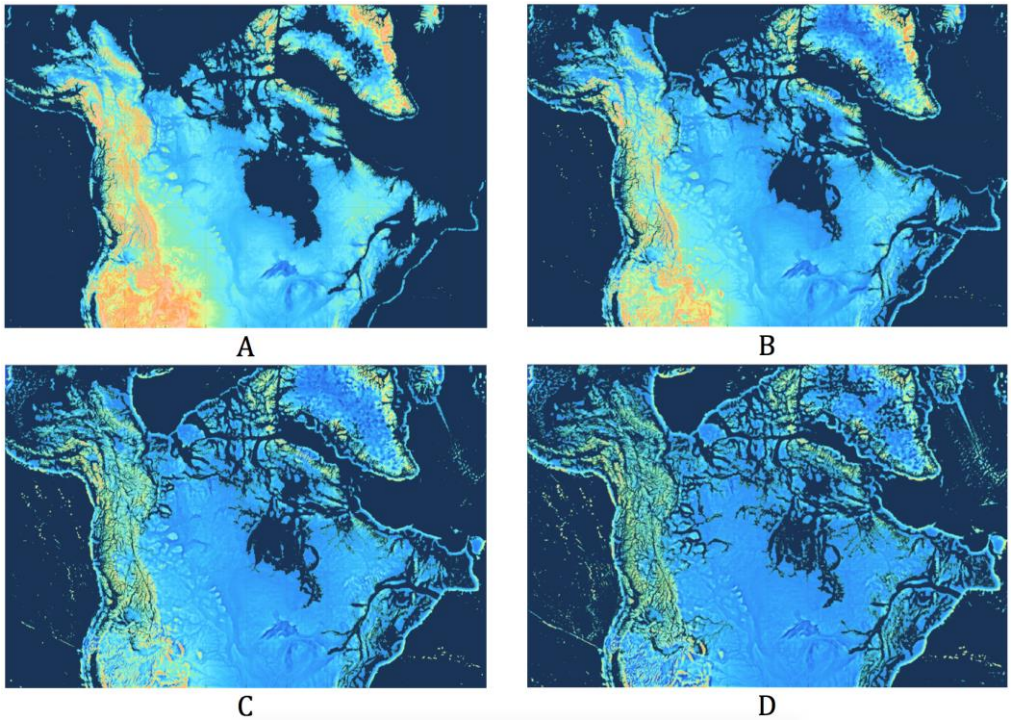

Figure 4: Land/ocean-mask for the same region, with the topography passed through a high-pass filter. Panel A: 40 % reduction in amplitude for wavelengths above 30km. Panel B: 60 % reduction. Panel C: 80 % reduction. Panel D: 90 % reduction. The ocean area increases slightly and the length of the coastline increases strongly.

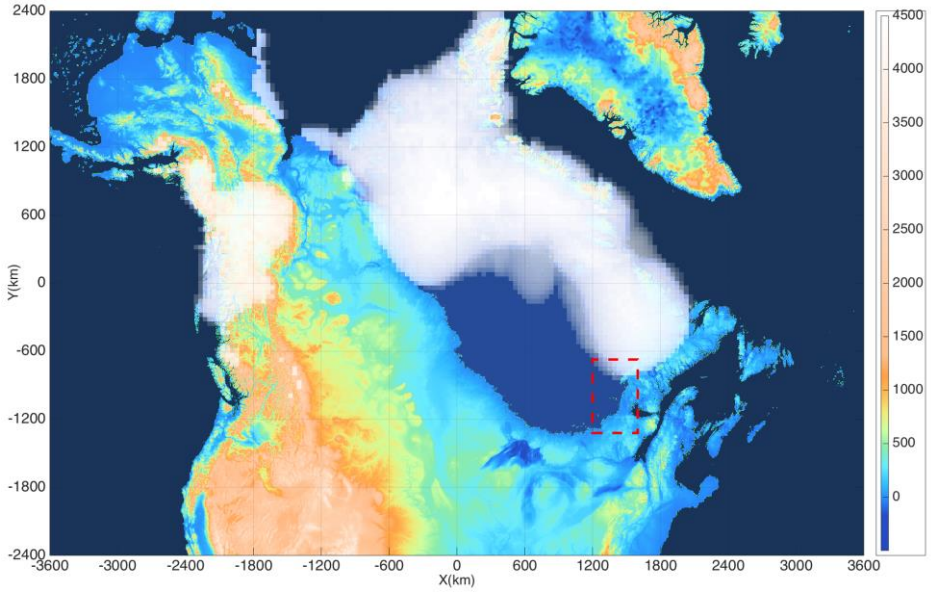

Figure 5: North America, the Laurentide ice-sheet and Lake Agassiz, 30ky before present. The red outlined region is shown in close-up in Fig. 6.

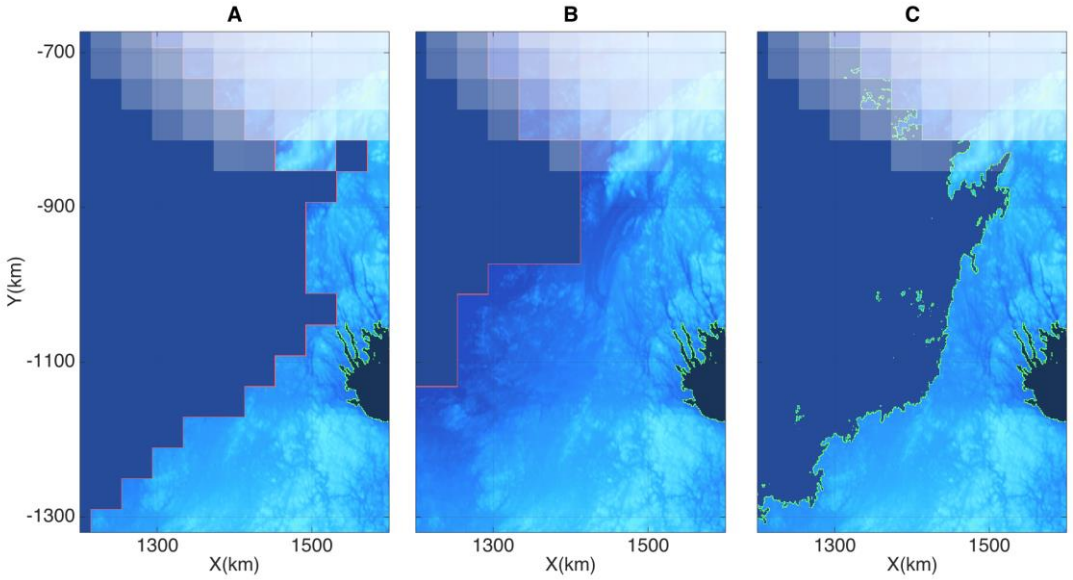

Figure 6: The area around the eastern shore of the lake, outlined in red in Fig. 5. Panel A: The 40 km resolution estimate. Panel B: The result of the 40 km maximum topography fill to a water depth 20 % below that of the 40 km estimate. Panel C: The final 1 km lake. Note that the algorithm allows for ice to float when the water column is high enough, creating a small ice-shelf along the margin, visible in panel C where part of the lake's shoreline runs beneath the ice.

Table 1: Computation time in seconds for stack and map conversion between different resolutions and for flood-filling at different resolutions with different starting stacks, for the 40 km - 8 km - 4 km - 2 km - 1 km series. The yellow shaded numbers are explained in the text.

Stack and map conversion						Flood-fill					
	40	**8**	**4**	**2**	**1**		**40**	**8**	**4**	**2**	**1**
40	-	0.04	0.04	0.07	0.20	**40**	0.05	0.19	0.85	3.73	27.54
8	-	-	0.34	0.37	0.56	**8**	-	0.94	0.16	0.76	11.65
4	-	-	-	1.30	31.22	**4**	-	-	4.03	0.40	8.29
2	-	-	-	-	65.98	**2**	-	-	-	16.89	5.64
1	-	-	-	-	-	**1**	-	-	-	-	82.04

Table 2: Computation time in seconds for stack and map conversion between different resolutions and for flood-filling at different resolutions with different starting stacks, for the 40 km - 20 km - 10 km - 5 km - 1 km series.

Stack and map conversion						Flood-fill					
	40	**20**	**10**	**5**	**1**		**40**	**20**	**10**	**5**	**1**
40	-	0.04	0.03	0.05	0.21	**40**	0.06	0.02	0.05	0.18	13.78
20	-	-	0.08	0.08	0.34	**20**	-	0.17	0.02	0.12	11.26
10	-	-	-	0.24	0.43	**10**	-	-	0.62	0.08	9.28
5	-	-	-	-	1.07	**5**	-	-	-	2.45	7.97
1	-	-	-	-	-	**1**	-	-	-	-	82.11

Table 3: Total computation time relative to the 1 km benchmark experiment for several resolution schemes, sorted ascending.

Resolution scheme					Computation time reduction (%)
40	-	-	5	1	88.64
40	8	-	-	1	84.79
-	8	4	-	1	50.13
-	8	-	2	1	10.25
-	-	4	2	1	5.80
-	-	-		1	0.00

Table 4: Reduction in computation time for the 40 km - 5 km - 1 km resolution scheme with respect to the 1 km benchmark experiment for different bedrock topographies. Also listed are the area and compactness factor of the resulting ocean basin.

Filter (%)	Compactness factor	Ocean area (km^2)	Computation time reduction (%)
-100*	6.0e-2	1.72e7	95.48
0	5.1e-3	1.72e7	91.56
20	2.7e-3	1.76e7	89.37
40	1.4e-3	1.77e7	74.48
60	7.4e-4	1.76e7	58.89
80	2.6e-4	1.79e7	35.35
90	1.4e-4	1.86e7	18.72

*30km low-pass filter; "smoothed" topography.