Publisher's note: the content of this comment was adjusted on 27 March 2026 after approval of the GMD executive editors since some formulations were inappropriate.

Dear editor,

sorry for taking such a long time and coming with little changes in the manuscript. I had to think whether or not it makes sense to proceed. [redacted] So I already feel that the extensions made in the first revision were mainly made for one reviewer with little overall improvement. Now I am running out of ideas what to change without destroying the paper completely.

[redacted]

Given the latest additions in cuPyNumerics, it could be beneficial to assess whether this library could circumvent the remaining issues with the Python implementation, and even tackle GPU acceleration.

Concerning "the remaining issues with the Python implementation", I was not aware that GMD no longer allows C or C++ codes. In the past, several publications in GMD even referred to models that were entirely implemented in C or C++. I am quite sure that Windows users will be able to copy the files minsia.py and pcg.dll into the same folder and that most of the Linux and Mac users will be able to compile a short C++ code with gcc or clang. Those who have no idea what compiling is are not my target group.

[redacted] Concerning the performance, I added a few results in the detailed response. Compared to my own solver, the standard SciPy solver (CG without preconditioning since the preconditioner is not available and is also sequential in its basic structure) was about 17 times slower (both normalized to a single CPU) in a typical setup. Since cuPyNumerics just introduces parallel implementations and no new algorithms, we will start from a loss by a factor of 17. To my experience with parallel computing (although not with GPUs), we will even not compensate the factor of 17 so easily in the CG solver since there is too much interaction in each iteration. Anyway, I will not buy a high-end NVIDIA just to proof that it makes no sense to speed up a bad algorithm.

The smooth-free reference should also be addressed, even if using a lower resolution reference.

Let me start with the "lower resolution reference". I guess that this point refers to the suggestion "A simple and computationally cheaper simulation using $\delta x = 1.0$ km without smoothing would be enough". I could do this and the reviewer...

... would be happy that the deviation between $f = 0.25$ and $f = 0$ is zero. Smoothing is performed over a length of $\pm fh$ into each direction, where h is ice thickness. At $f = 0.25$, this would be less than 160 m for the maximum ice thickness found here ($h < 640$ m). So there will be no smoothing at $\delta x = 1$ km and the results for $f = 0.25$ will be "exact" [redacted]

Just to recapitulate: My original "convergence analysis" was based on the usual approach with a geometric series. If we compare, e.g., the deviations $|u(1) - u(0.5)|$, $|u(0.5) - u(0.25)|$, ... (argument refers to the value of f) and find $|u(f) - u(0.5f)| < \lambda |u(2f) - u(f)|$ with $\lambda < 1$ for each f ($\lambda = 0.5$ = linear convergence), we can use a geometric series

$$\begin{aligned} |u(0.25) - u(0)| &\leq |u(0.25) - u(0.125)| + |u(0.125) - u(0.0625)| + \ldots \\ &\leq |u(0.5) - u(0.25)| \left(\lambda + \lambda^2 + \ldots\right) = |u(0.5) - u(0.25)| \frac{\lambda}{1-\lambda} \end{aligned}$$

The results from Fig. 2 already suggested $\lambda < 0.5$ (so faster convergence than linear), which means that the deviation between $f = 0.25$ and $f = 0$ is smaller than the deviation between $f = 0.5$ and $f = 0.25$.

The reviewer(s) did not believe in this [redacted]. I added a shorter simulation over 100 yr that exactly shows this result (Fig. 4). Of course, this simulation did not start at $t = 0$ because the effect of smoothing is very small initially, but at $t = 2900$ yr (to 3000 yr) in order to get the strongest effect.

[redacted] Of course, it is the deviation between the approximation ($f = 0.25$) and the "exact" solution ($f = 0$) over a 100-year time span only, starting from a given initial state. It does not matter where the initial state is coming from; it might even be a measured ice thickness. So even if [redacted] the "strong bias" claimed by the reviewer was true, it would have no effect on the analysis. The error over the 100-year time span is, of course, smaller than over 3000 years. Not 30 times smaller, but only about 3 times smaller here because the effect of smoothing is weak in the beginning. But is is only to show that the deviation between $f = 0.25$ and $f = 0$ is really smaller than the deviation between $f = 0.5$ and $f = 0.25$.

Sorry for writing such a long text. [redacted]

Motivation could be improved according to the reviewer's suggestions. From previous discussion it seems clear that limitations of SIA are known and should therefore be accurately acknowledged within the manuscript, find the appropriate way of improving this.

After reading the introduction again several time, I still feel that it addresses the limitations of the SIA sufficiently, including the references suggested by Thomas Zwinger. It seems that the reviewer still wants me to remove the sentence "The purpose of this paper is to challenge the hypothesis that the potential for further improvements in computational efficiency with classical numerical methods is limited (Jouvet et al., 2021)". However, I was motivated by the feeling the there has been no progress concerning numerical performance for several years – not because the schemes were already so good as claimed by Jouvet et al. So I will not remove this sentence.
It seems to me that the reviewer misinterprets this motivation in such a way that MinSIA could compete to the IGM. I clarified this in the conclusions, but I feel that the limitations of the SIA have been addressed sufficiently.

I encourage the author to carefully address the suggestions raised by the second round of revision as those would further significantly improve the reach and exposure of this work.

I am afraid that it will be limited and not depend much on whether I am willing to follow the weird suggestions of the reviewer, although the number of downloads from the repository is not too bad. Anyway, the two reviewers motivated me to work on the physical limitations of the SIA with focus on valleys. This is going better than I thought. So there will be a subsequent manuscript building up the numerics proposed here, which will hopefully be accepted better by the community. In this sense, it was worth the effort, no matter what happens to the present manuscript.

Note that "major revision" are chosen in order to keep the opportunity to send out a revised version of this manuscript draft to external referees.

I would be happy about a reviewer with a bit more background in numerics. On the other hand, my motivation to consider completely different new concerns has already decreased.

Best regards,
Stefan Hergarten

Reviewer 1 (Daniel Moreno-Parada)

Unresolved issues

Parallel computing in Python (cuPyNumeric). Since the paper is motivated to show that there is still room for improvement in computational efficiency with classical numerical methods, I consider necessary to include a section dedicated to parallel performance. Given that the Python version of MinSIA is currently available, my comment in the previous report becomes even more relevant: NVIDIA recently announced the cu-PyNumeric library (https://developer.nvidia.com/cupynumeric), a drop-in replacement for the NumPy library that allows to run on multi-core CPUs, single or multi-GPU nodes, and even multi-node clusters without changing your Python code. Operations are executed by Legate's task engine and accelerated on one or many NVIDIA GPUs (if no GPU is present, on all CPU cores).

I am afraid that we will never agree on this point. [redacted] Some test results for the typical $\delta 4 = 0.25$ yr from $t = 3000$ yr to 3010 yr:
(1) pre-compiled PGG with preconditioner (≈ 12 iterations per step): 117 s CPU time, 117 s wall clock time at 1 CPU
(2) SciPy CG solver without preconditioner (> 300 iterations per step): 1973 s CPU time, 555 s wall clock time at up to 4 CPUs
So SciPy without the preconditioner (which is essentially sequential) starts at a disadvantage by a factor of 17 compared to the precompiled solver. To my experience, we will hardly compensate this disadvantage for differential equation problems even with a large number of GPU cores (in contrast to some data analysis and AI tasks).

Linking with the previous comment, as the author justify the absence of smooth-free reference simulations as a results of prohibiting computational costs, this apparent issue could be potentially overcome by using cuNumeric library. The library is extremely well documented (complete user guide) and straightforward to install: https://docs.nvidia.com/cupynumeric/latest/installation.html.

Going back to $\delta t = \frac{1}{4096}$ yr, the disadvantage of the SciPy solver decreases apparently:
(1) pre-compiled PGG with preconditioner (1 iteration per step): 31408 s CPU time, 31410 s wall clock time at 1 CPU
(2) SciPy CG solver without preconditioner (≈ 2 iterations per step): 72734 s CPU time, 44340 s wall clock time at up to 4 CPUs
However, the CG solver does not contribute so much here, so that only 1.6 CPUs are used on average. So a GPU will probably not be much better here. [redacted]

Missing smooth-free reference. In the revised version, the author additionally includes a 100-year-long smooth-free run starting from a 2900-year-long smoothed simulation ($f = 0.25$). The reference $f = 0.25$ is then considered to be valid by showing that the deviation of the 100-year-long smooth-free run is smaller than the smallest deviation compared to larger smoothing factors. I find this experiment lacking since the smoothed solution (initial condition) already introduces a strong bias. A 100-year-long simulation starting from an already smoothed solution (i.e., 2900-year-long running with $f = 0.25$) does not say anything about the reliability of the method in the first place.

It is the deviation between the approximation ($f = 0.25$) and the "exact" solution ($f = 0$) over a 100-year time span, starting from a given initial state. It does not matter where the initial state is coming from; it might even be a measured ice thickness. So even if the [redacted] "strong bias" claimed by the reviewer was true, it would have no effect on the analysis. The error over the 100-year time span is, of course, smaller than over the full 3000-year time span. Not 30 times smaller, but only about 3 times smaller here because the effect of smoothing is weak in the beginning and the topography is already not so far away from a steady state, which limits the growth of the error through time.

I completely agree with the previous report of Reviewer 2 (Thomas Zwinger): "I do not think that comparison to a result obtained with the method is able to show the accuracy of the method itself". The author argued that the limitation in computing time prohibits such a comparison. This can be simply overcome by performing a lower resolution reference test and then showing the implications of the method referred to a smooth-free reference (or enabling parallel computing, as mentioned in my previous report). Using the method itself does not test the validity of the solution, it solely explores the sensitivity to certain parameters and resolutions. This lower resolution experiment would additionally serve to test the hypothesis of Reviewer 2: "[...] this smoothing algorithm works because it re-introduces the basic principle of thin-film approximations to resolve horizontal gradients with respect to a large aspect ratio". A simple and computationally cheaper simulation using $\delta x = 1.0$ km without smoothing would be enough.

[redacted]
Smoothing is performed over a length of $\pm fh$ into each direction, where h is ice thickness. At $f = 0.25$, this would be less than 160 m for the maximum ice thickness found here ($h < 640$ m). So there will be no smoothing at $\delta x = 1$ km and the results for $f = 0.25$ and $f = 0$ will be exactly the same.

Motivation. The updated version still reads: "The purpose of this paper is to challenge the hypothesis that the potential for further improvements in computational efficiency with classical numerical methods is limited (Jouvet et al., 2021)". It must be stressed that the work of Jouvet et al, (2021) employs PISM output to train the emulator, capturing the physics contained in longitudinal stresses present in the SSA. While the results are presented as an improvement in terms of classical numerics, I do not consider them to be a comparable alternative to the work of Jouvet et al, (2021).

Right – this was the original motivation behind the development, and the sentence will stay there. I never said that the simple SIA-based approach can compete with the IGM. I clarified it in the conclusions, [redacted]

Moreover, how does the present smoothing method compare to Fig 7.4 in Greve and Blatter (2004)? In my previous report, I highlighted the problematic SIA stress balance at such extremely high resolutions. Already described by Greve and Blatter (2004), the SIA is only justified for large ice sheets where conditions generally vary little over horizontal distances (i.e., 5-10 times the local ice thickness). However, this is not the case in mountain glaciers, where longitudinal and transverse coupling of stresses are important and should therefore be considered.

As already written in my first response, the smoothing method is completely independent of the physical limitations of the SIA. I still believe that Fig 7.4 in Greve and Blatter (2004) illustrates the deficiencies of the SIA quite well, but what should I do with this figure?

Lack of references to prior work. There are some important works neither referenced nor discussed. For instance, Cheng et al. (2017) introduced an adaptive time step control for simulations of the evolution of ice sheets using Elmer/Ice (Gagliardini et al., 2013). Semi-implicit and fully implicit methods are compared for a number of discretization stencils, closely related to Sections 4.2 and 5. I consider that if the problem of "finding the best time increment" is to be tackled, the paper should dive into the predictor-corrector (among others) approaches and showcase the performance in MinSIA.

We will never agree on the paper of Cheng et al. (2017). As explained in my previous response, a fully implicit scheme that is still limited by the Fourier criterion is very little progress. And I do not find it noteworthy that an adaptive time-stepping scheme can improve the performance. I will keep the paper in mind as a good example for the little progress made in the previous years and also a for the possibility to publish everything as long as the formalism is complicated enough. However, I will never cite it.

As stated in my previous review, the paper may also benefit from merging Section 4.2 (The maximum time increment) and 5 (Finding the best time increment) for clarity. All numerical results regarding time-stepping should be described and discussed therein.

I already disagreed to the suggestion in my first response.

Other references are still missing in the current version. For instance, the second-order SIA performance in this context (e.g., Ahlkrona et al, 2013).

I have no distinct opinion about second-order SIA (including the iSOSIA) and also not about the combination with the SSA in PISM. The results I have seen in tests do not look bad, but I am not fully convinced that they are really good in alpine valleys. Furthermore, I have no idea which opinion the reviewer wants to promote here.

Minor comments

I have run the Python code and it creates one file per frame to save the relevant data. This eventually leads to a large number of files for each simulation. It would be optimal have one file per run where all data frames are stored.

MinSIA is not a model to be run "out of the box". In particular, data handling is left to the user. The file simul.py just reflects the way which I find useful (being able to view intermediate results, not having to append data to a file, being able to delete time slices that I do not need), which is opposite to the opinion of the reviewer.

The solver is written in C++ and must be compiled beforehand. In line with the first unresolved issue, it would be beneficial to parallelize the solver in Python using e.g. cuPyNumeric and discuss model performance.

As written above, I do not believe that a parallel version based on the respective solvers available in SciPy will reach the performance of the version with the C++ solver on a single core. [redacted]