Reviewer 2 (RC2)

The article details a plugin functionality added to the ICON model, based on a common adaptor library to be used by both the ICON model and the plugins. Entry point hooks are inserted in the ICON model to execute the plugins at various places. The authors introduce the concepts in a structured manner comparing all possible approaches to extend models with external components. The described code is open-source and readily available to explore, including documentation and examples. Various conceivable scenarios are considered in how plugins operate with the given ICON data fields. The article fits well in the context of this journal and is of high quality. I therefore recommend it for publication.

One question. Given that it is possible for plugins to share the data pointers with ICON, is there an optional safety mechanism provided (e.g. by checksumming the data) in order to ensure or verify that no data has been modified inadvertently?

Thank you for this positive evaluation of our work.

At the moment no such safety mechanism is provided, as ComIn is operating with Fortran pointers and memory access restrictions are not implemented in the language standard. The access information provided in ComIn when receiving a data pointer will be used in the future on the host side for synchronisation operations to provide updated variable fields via the pointers according to the requested access patterns. However, a possible extension of ComIn might use the information if a data field is obtained only to read or also to write and throw a warning in the former case if the data is indeed changed. As these checks would add an overhead on a simulation, this option would only be available in a debug mode, not in the standard configuration. We will provide this information in the manuscript and extend Section 3.5 accordingly (changes/updates are given in bold font below).

"The implementation of how the ICON **data fields and** descriptive data are made available to ComIn impacts the overhead and usability of ComIn. The smallest memory overhead is achieved when sharing pointers to the ICON memory addresses. However, this adds the possibility that a plugin can (inadvertently **or incorrectly) change the value of such a field. Via the usage of Fortran pointers, no access restrictions are implemented, as these are not supported by the language standard. Implementing memory access restrictions would add additional undesired execution overhead. It**

was thus decided to share pointers to data fields via ComIn without safety mechanisms.

So far the memory argument (ComIn should be lightweight) was considered more important also in the case of descriptive data, even though most descriptive data are constant in time and their size not excessive. Thus, with a few exceptions, pointers to the ICON descriptive data are shared directly. Another advantage of this approach is that inconsistencies in data copies can be prevented. Exceptions include the cell properties longitude and latitude, as their storage format has been simplified in ComIn to eliminate a further dependence on the host model's data structures. It is thus the responsibility of the user to ensure that the descriptive data are handled correctly and that descriptive data are not accidentally modified."

We have also updated the description of the associated point in Section 6 (line 480 in the old version of the manuscript):

"When a plugin requests an ICON data field it does this for a specific list of entry points. This information can be used by ICON to determine the exchange of data throughout the simulation. With the additional information of the desired access, i.e. if data is just read, adapted by a plugin or also required for halo synchronization, the host model could detect incorrect access patterns **and return an error or a warning if access restrictions are violated**. The latter part is currently not implemented but a potential expansion for the future, probably available only in a debug mode of ComIn **as this option would add an overhead**. Additionally, this information can be used to support asynchronous execution of plugins and to enable efficient halo synchronization through ComIn."