Use of general purpose graphical processing units with MODFLOW-2005 (Ground Water Manuscript GW20120518-0099)

Authors: Hughes, J.D., and White, J.T

September 4, 2012

We greatly appreciate the comments provided by the three reviewers and the editor. We have addressed all of the comments in this document. In general, we have made all of the suggested modifications. In a few cases we disagreed with reviewer comments but have provided a justification for not accepting the comment.

Reviewer comments are included (*in italics*) prior to our responses. Revised material in the manuscript is indicated using blue text. Text indicated in cyan in the revised manuscript represent changes in response to internal USGS reviewers.

Editor's Comments:

1. *Our layout doesn't require section numbering, so please delete those.*

   Section numbering has been eliminated in the revised manuscript.

Reviewer 1 Comments:

1. *The introduction should also introduce the speedup potential and bottlenecks of GPGPU and parallel CPU processing for the UPCG solver.*

   Additional discussion of ideal parallelization and issues that make achieving ideal speedup has been added to the introduction.

2. *Section 2 has too much jargon. I suggest removing what is irrelevant and explaining what is, in terms of how that helps speed up or slow down GPGPU processing compared to serial computing or multi-core processing on CPUs.*

   We have attempted to eliminate all but the essential GPGPU jargon in the main text. A significant portion of the jargon has been moved to Boxes 1 and 2.

3. *The manuscript needs to mention how the solver is available for M2K5 for the GPGPU and multi-core CPU versions (are they DLLs, how are they linked into the main code, etc). I do not expect people to generally compile their codes, let alone within CUDA. Some direction therefore is needed on accessibility.*

A section titled "**How to Obtain the Software**" has been added that provides information on how to obtain the software. The source code and a precompiled Windows executable will be provided. Instructions for compiling the source code have been provided in the "**How to Obtain the Software**" section.

4. *Also, direction is needed on hardware that may be best suited for the GPUGPU processing will benefit this manuscript.*

   The authors are not experts on the hardware that is best suited for GPGPU processing. Rather than providing guidance on appropriate hardware we feel it is more appropriate for potential users to consult with a qualified software dealer and identify the hardware that is most appropriate for their particular organization.

5. *GPGPU speedup has been compared to the PCG5 solver on a serial CPU. With more advanced solvers coming into the MODFLOW family (for example the XMD solver in MODFLOW-NWT), it would be interesting to note the speedup of UPCG compared to even more robust higher level ILU solvers which may be even faster than PCG5 for hard problems.*

   The authors agree with the reviewer that higher level ILU preconditioners may be more effective than the standard MODFLOW PCG solver for particular types of problems. The analyses presented in this manuscript are not meant to be an exhaustive comparison of GPGPU solutions but an evaluation of the speedup possible when compared to a standard MODFLOW solver that uses a similar solution approach.

6. *Issues with GPGPU processing for different schemes are scattered throughout the manuscript. These issues should be collated into a separate section or a table. For instance: GPGPU memory requirements for larger problems; need to maximize the amount of time spent in the linear solvers; the overhead associated with GPGPU-CPU memory copy operations relative to the solution time; higher percentage of runtime spent transferring preconditioner data; why the total time spent applying the preconditioner actually increases for the MILU0 preconditioner; what is less optimized in Intel FORTRAN for multi-core processing; details on problem size limitation due to RAM available on the GPGPU; and others.*

   The authors disagree with this comment. We believe the issues related to solution of the groundwater flow equation on a GPGPU are presented in the relevant locations and important issues are summarize again in the "**Conclusions**" section.

Reviewer 2 Comments:

1. *Overall, lots of jargon is used. Naturally, this is an emerging field, and one that many readers of Ground Water will not be familiar with, so the use of jargon will be difficult for the uninitiated reader. For example, lines 75-79 are filled with terms that will mean nothing to the average groundwater modeler - consider deleting (or moving to appendix*

or online material) this paragraph and other similar paragraphs that might not hold much meaning to the average reader.

We have tried to eliminate all non-essential GPGPU jargon and make the main text more readable for the typical Ground Water reader. A significant portion of the jargon has been moved to Boxes 1 and 2 as suggested by the editor.

2. *Thorough comparisons were made for different preconditioners and different parallelization strategies and the authors did a great job of making the comparisons fair. Considering the results of the various preconditioners, it appears that Jacobi preconditioning is good enough for many of the problems, which leads me to believe that the K fields are fairly smooth (and the spread of the eigenvalues are nicely clustered). I tend to think of this as the exception in rather than the rule in most practical groundwater models. Can you provide guidance for using MIC vs. POLY vs. Jacobi based on the K field? In other words, I think these results are highly dependent on the K field and only 2 fields are used. How would the results vary if you stretched the contrasts in your K field? In that regard, to be fair, the GMG solver should also be used as a basis for comparison and give insight about which strategy should be used for optimal performance.*

   Evaluation of the maximum and minimum eigenvalues for the original 1000 $\times$ 1000 problem suggests that there is clustering of eigenvalues in the original $\mathbf{A}$ matrix. It was not possible to evaluate the full spread of the eigenvalues because of the size of the $\mathbf{A}$ matrix. Evaluation of the eigenvalues of the final preconditioned matrix for the Jacobi preconditioner, however, indicates there are two clusters with the first ranging from approximately 2.5×10^{-4} to 1×10^{-2} and approximately 1.5×10^{-2}.

   To look at the effect that eigenvalue clustering has on the effectiveness of the Jacobi preconditioner we have evaluated a number of alternative problems in an effort to reduce the smoothness of the fields. The additional fields that were evaluated are based on the fields used to evaluate the performance of the UPCG solver presented in the manuscript and included: (1) random permutations values in the original fields; (2) fields that used the average hydraulic conductivity in select quantiles of the original fields (to develop a zoned field); (3) a constant field based on the mean hydraulic conductivity of the original fields; and (4) a stochastic field that used the same properties of the original fields except with a variance that varied over 5 orders of magnitude (the original varied over 3 orders of magnitude). Evaluation of the eigenvalues of the final preconditioned matrix for the Jacobi preconditioner indicates that the eigenvalues for (1) are clustered in three groups with a cluster covering a large range from approximately 5×10^{-4} to 7.5×10^{-3}, (2) show no visible signs of clustering, (3) show no visible signs of clustering, and (4) are clustered into two groups like the original field but span a larger range.

   For all of these alternative fields, the Jacobi preconditioner was able to solve the problem and required a comparable number of iterations (approximately 4,500 iterations for the 1,000 $\times$ 1,000 $\times$ 1 problem) except for the fields with a larger variance (approximately 5,200 iterations for the 1,000 $\times$ 1,000 $\times$ 1 problem). The authors suspect the Jacobi preconditioner is sufficient in almost every case evaluated because of the

number of iterations that are allowed (50,000). In fact, we were able to get the 4,000 $\times$ 4,000 $\times$ 1 problem to solve if the maximum number of iterations was increased (a solution was achieved in 66,861 iterations).

We have not included results for any of these additional simulations but have added additional discussion of how speedup may be affected by the hydraulic conductivity field has been added to the **Discussion** and **Conclusions** sections.

3. *There seems to be a tradeoff between parallelizing on the CPU vs. the GPU depending on the sophistication of the preconditioner. What is not clear was if it Open-MP was used on PCG2. Reference is made to the work by to Dong and Li, but it appears that this was not considered. How do these results compare to that of Dong and Li? The intent here is not to say that your results are better/worse than those of Dong and Li. Simply, not everyone has access to a video card with double precision GPU capabilities, whereas multicore CPU's seem to be more common, so some guidance of which strategy might be more effective depending on the user's hardware would be helpful (ie, If I were buying a new workstation, should I focus on getting the best multicore CPU, or should I get a less powerful CPU and spend money on a high-end GPU?) I understand that the answer to this question is not straightforward, but any additional insights you could provide regarding this tradeoff would be helpful.*

   OpenMP was not evaluated on PCG2 but was considered using the UPCG solver. We believe we can accurately report on conditions under which use of a GPGPU solution would be preferred to an OpenMP solution and vice-versa based on (1) the similarity of the effectiveness of the UPCG solver with the MILU0 preconditioner and the PCG solver with the MIC preconditioner and (2) the fact that UPCG solver speedups with OpenMP are comparable to the Intel compiler results reported in Dong and Li (2009) . We have revised the **Conclusions** section (see response to **Reviewer 2 Comment 2**). We have also evaluated the effect that Intel compiler optimization has on OpenMP speedups and have revised the **Discussion** section to reflect the observation that speedups were comparable with and without use of the -O3 compiler switch.

   We have also included our assessment of the best approach for being able to run a range of MODFLOW problems in parallel has been added to the **Conclusions** section.

4. *Line 26 - "can be *easily* parallelize"*

   Suggested addition made.

5. *Lines 28 - 34 - there is some time discrepancies here from 2005 to 1984, to 1988. Perhaps just say "MODFLOW" on line 28 and remove the "since 1984" and insert "numerous" before "two-and"*

   Suggested modification made. Also MODFLOW-2005 has been replaced with MODFLOW in the revised manuscript except where it is necessary to specifically discuss the current version of MODFLOW.

6. *Line 47 "present" (plural)*

Correction made.

7. *Line 48 - unclear - faster at what exactly? Operations,cycles/second?*

   This sentence has been revised to indicate that GPGPUs are capable of executing more floating point operations per second than CPUs.

8. *Lines 75-108 is filled with a lot of technical jargon. Consider moving some of this to an appendix (as noted above).*

   A significant portion of the jargon has been moved to Boxes 1 and 2.

9. *Line 108 - really, the parallelization you mention can be applied to many simulators, not just MODFLOW-2005. Consider mentioning wider range applicability.*

   We have added text to indicate that GPGPU parallelization has great potential for MODFLOW and other numerical simulators.

10. *Line 213-214 - unclear. Consider rephrasing*

    This sentence has been rephrased to clarify the description of the thickness of layer 1.

11. *Line 351 - "developed to *use* the PCG" (one does not "solve" the algorithm)*

    Suggested modification made.

12. *Line 376 - as indicated above, can you provide some guidance here about when one approach might be better than another depending on the system properties (size, K field). Also, GMG should be included here to so that the full suite of approaches are compared (weak but highly parallelizable to strong but not very parallelizable).*

    See response to **Reviewer 2 Comment 2**.

13. *Figure 4 - I think this figure is extremely important as it shows where the speed ups occur. However, it is not clear how much time is spent in the GPU for the matrix-vector and BLAS operations because the bars seem to not show up on the scale? Can you include a value here so the reader does not assume these are zero.*

    Figure 4 has been modified as suggested.

14. *In light of the speed ups coming from CUBLAS operations being parallelized over the cells in the model, I am wondering if additional speed up could be obtained by using other primitives in the CUSparse library for the triangular solves and/or the decomposition? Was this considered as well? There is some new work that tries to parallelize ILU (see: http://research.nvidia.com/publication/parallel-incomplete-lu-and-cholesky-factorization-preconditioned-iterative-methods-gpu)*

    We are aware of this work but have not attempted to implement it. One of the reasons we did not devote any time to its implementation is Naumov (2011) only reported an average speedup of 2.0 which is less than was observed with the GLSPOLY preconditioner. The work of Li and Saad (2010) pointed us in the direction of the GLSPOLY preconditioner. However, we have included a few sentences discussing recent

work devoted to improving the effectiveness of incomplete factorization preconditioners in the **Conclusions** section.

Reviewer 3 Comments:

1. *Quality of figures should be improved (all figures but in particular, figures 3 to 6). Texts (titles and legends) are too small to read. Lines are very hard to read.*

   The font sizes have been increased on all of the figures. The symbol size and lines on **figures 3**, **5**, and **6** have been increased and converted to solid colored lines, respectively, to improve the readability as suggested.

2. *Tables should be reworked. I do not think we need 3 digits after decimal point for these comparisons. Please also consider significant digits in the tables.*

   Double-precision floating point variables are used in the solvers and the tables have been rounded to single-precision numbers. Since we are reporting results from a numerical analysis we thought it appropriate to retain a larger number of digits to allow readers the ability to evaluate our results differently than we have. For example it would be possible for readers to calculate accurate run times (in seconds) for each of the simulations summarized in **table 2** using the speedup values and MIC cpu times. This may be of benefit for future studies conducted by other researchers. However, we would be willing to modify the number of digits reported if the editor agrees with this comment.

3. *The authors only consider total execution time which includes time required for both Picard and linear solver iterations. Did you closely examined these iteration performance in solver comparisons? How these iteration performance changes among different preconditioning schemes?*

   A table has been added (revised **Table 1**) which includes information on the number of iterations required for each of the solvers and various preconditioners. Additional discussion has been added in the **Discussion** section to provide some guidance on the strength of the preconditioners for the problems evaluated.

4. *Lines 28-32: the authors mentioned computational performance changes due to domain discretization. Does this cause by FD formulation or matrix solver scheme? Explanation is needed.*

   This is primarily based on the increase in the number of simultaneous linear equations being solved. Differences in the **A** matrix resulting from different cell sizes can also affect solver performance. Additional clarification has been added.

5. *Line 48: 448 → 448 cores.*

   Suggested addition made.

6. *Line 114: is L3/L3T correct?*

Yes. See **equation 2-2** in Harbaugh (2005).

7. *Line 123: is L2/T correct?*

   Yes. See **equation 2-9** in Harbaugh (2005).

8. *Line 161: What is CUBLAS?*

   CUBLAS and BLAS have been defined.

9. *Line 199: Problem Description. Reasoning for choosing test cases is helpful for the readers.*

   Additional discussion of the reasons the particular test cases were evaluated has been provided.

10. *Line 236: "(10-0)/1000"is not necessary.*

    Suggested modification made.