Geosci. Model Dev. Discuss.,
https://doi.org/10.5194/gmd-2020-183-AC1, 2020

Geoscientific Model Development Discussions

Interactive comment on "Fast and efficient MATLAB-based MPM solver (fMPMM-solver v1.0)" *by* Emmanuel Wyser et al.

Emmanuel Wyser et al.

emmanuel.wyser@unil.ch

Received and published: 14 August 2020

First, we would like to thank the referee for the time spent on reviewing our paper and for his stimulating comments.

Referee's comment 1

The paper presents an explicit vectorised form of the material point method for the generalised interpolation and CPDI2 variants of the method. The paper is reasonably written but is missing many details on the implemented algorithms and the numerical analyses are too focused on reproducing the results of others rather than on numerical performance which is the main trust of the article.

Author's reply 1

Thank you for the time spent on reviewing our work. We acknowledge the lack of details on the implemented algorithm. We will develop the numerical implementation in the revised manuscript. In addition, it is true that the numerical analyses are more focused on reproducing results of other studies. Therefore, we will investigate further the numerical performance of the solver in the revised manuscript, i.e., a detail analysis of performance will be presented for a wide range of numerical setup (increasing mesh resolutions, initial number of material points per element, etc.). This should provide a more exhaustive analysis of the numerical performance.

Referee's comment 2

The introduction to the paper appears to have picked a random selection of MPM articles rather than focusing on articles that look at the numerical implementation of the method. The introduction should be made more coherent and focused.

Author's reply 2

We acknowledge the lack of coherence of the introduction section. Consequently, we will focus our introduction toward the numerical implementation of MPM and add references which focus on numerical implementation of MPM and FEM, since both share common grounds.

Referee's comment 3

The authors have not picked up on any of the papers that extend the MILAMIN approach. In particular the authors should refer to the following paper that extends the MILAMIN ideas to non-linear problems: O'Sullivan, S, Bird, R.E., Coombs, W.M. & Giani, S. (2019). Rapid non-linear finite element analysis of continuous and discontinuous Galerkin methods in MATLAB. Computers and Mathematics with Applications 78(9): 3007-3026. There are others and the authors should review the appropriate literature.

Author's reply 3

Indeed, we were not aware of further extension to the MILAMIN approach. We reviewed the reference proposed and found common approach concerning vectorization, i.e., the use of the accumarray function to accumulate internal force contributions to nodes. Additionally, we will review the appropriate literature and consequently update the introduction section.

Referee's comment 4

If performance is the focus, why use MATLAB? Why not adopt one of the existing MPM codes that are written in compiled code which will always be faster than MATLAB? Such as: jixiefx.com, github.com/yuanminghu/taichi_mpm, cimne.com/kratos/, github.com/nairnj/nairn-mpm-fea, github.com/cbgeo/mpm/, sourceforge.net/p/mpmgimp, github.com/xzhang66/MPM3D-F90. There may be others.

Author's reply 4

It is true that MPM codes written in compiled codes will always be faster than MATLAB. Our point is to show that one can use MATLAB to produce prototyped code with a decent computational performance. However, our true concern is to propose an efficient and vectorised MATLAB code, which could be easily translated into the C CUDA language, or, at least, which partially minimizes the hurdles of the syntax translation while preserving the algorithm. As such, we will clearly state such concern in our revised manuscript.

Referee's comment 5

The authors mention an implicit implementation but do not present any results in terms of the speed gains of the implicit vectorised algorithm. These should be included and the vectorised algorithm explained.

Author's reply 5

It is true that i) we do not present any results related to speed gain and, ii) the implicit implementation is not explained. Our concern was to show that the error saturation

we reported for the explicit implementation was due to the explicit formulation. Consequently, an implicit implementation under the vectorisation framework of the explicit formulation should converge without any saturation error. We showed it did. However, we can include in the Supplementary Material of the paper a description of the vectorization for the implicit implementation if needed.

Referee's comment 6

CPDI2 methods suffer from issues associated with domain distortion and are not suitable for problems involving large shear/rotation. This point should be acknowledged, see for example: Wang, L., Coombs, W.M. , Augarde, C.E. , Cortis, M. Charlton, T.J. , Brown, M.J. Knappett, J., Brennan, A. Davidson, C. Richards, D. & Blake, A. (2019). On the use of domain-based material point methods for problems involving large distortion. Computer Methods in Applied Mechanics and Engineering 355: 1003-1025.

Author's reply 6

It is true that CPDI2q suffers from domain distortion, as shown in the cited reference by the referee. We will clarify this in the revised manuscript. In general, we will discuss in greater details advantages and flaws of domain-based MPM.

Referee's comment 7

What large deformation formulation has been used in this paper in terms of stresses and strains? Logarithmic strains and Kirchhoff stresses? Explain and justify the large deformation framework.

Author's reply 7

The large deformation formulation is based on the infinitesimal strain coupled to an objective measure of the stress for the finite deformation, i.e., the Jaumann stress rate formulation. It is a widely accepted deformation framework in a number a study, see Bandara, S., Ferrari, A., Laloui, L. Modelling landslides in unsaturated slopes subjected to rainfall infiltration using material point method. International Journal for Numerical and Analytical Methods in Geomechanics. 40, 1358-1380 and many others. We think for an explicit implementation with small time step, such deformation formulation stands to reason. These considerations will be specified in the revised paper and the large deformation framework we selected will be clearly stated.

Referee's comment 8

How has the plasticity algorithm been implemented within the large deformation framework? The authors mention a prediction/correction type algorithm so how have they recovered the additive decomposition of the elastic and plastic strains from the multiplicative decomposition of the deformation gradient? Critical details are missing here.

Author's reply 8

It is true that critical details are missing concerning the plasticity algorithm we selected. At first, we did not want to emphasize return algorithm used in this study. But since the concern of the referee, we will add more details concerning the way plasticity is handle in the revised manuscript. More specifically, the prediction/correction algorithm used in this study combined with the small strain theory and the objective stress measure allow us to afford the decomposition of the elastic and plastic strain from the decomposition of the deformation gradient, as done in the MATLAB code AMPLE. We preferred the approach of Huang, P., Li, S-L., Guo, H., Hao, Z-M. Large deformation failure analysis of the soil slope based on the material point method. Computational Geosciences. 19. 951-963.

Referee's comment 9

How are boundary conditions imposed in this model?

Author's reply 9

Boundary conditions can be difficult to impose thoroughly in MPM, especially when considering non-aligned boundary conditions with respect to the background mesh, as mentioned in Cortis, M., Coombs, W., Augarde, C., Brown, M., Brennan, A., Robinson,

S. Imposition of essential boundary conditions in the material point method. International Journal for Numerical Methods in Engineering. 113: 130-152. This is the main reason why we choose numerical problem and setup in which boundary conditions could be directly applied on the background mesh. We will mention this important point of aligned boundary conditions in the revised manuscript.

Referee's comment 10

Figure 6 - what causes the step in the red line around 10ËĘ3 material points? Is there a shift in terms of the cost within the algorithm?

Author's reply 10

Thank you for this comment. We started an investigation in terms of FLOPS and L2 cache concerning this step around 10ˆ3 material points and we just concluded this shift is due to the overhead and RAM-to-cache transfer. As mentioned further by the referee, overheads are a major concern in MATLAB. This step typically occurs when the block of data send from RAM to the L2 cache exceeds its maximum capacity. MATLAB can longer treat information as a contiguous block of data sent one time and start swapping from RAM to cache and back and forth. This costs an additional computational expense which results in this step in Figure 6 in the paper (see FIG. I).

FIG I: a) Floating-point operation per second in million and, b) linear increase of RAM usage with an increasing number of material point. The limit when the allocated space in RAM reaches the L2 cache size (1024kB) corresponds to the drop of FLOPS in FIG I a). Thanks to the referee's comment, we now come with a different approach to vectorise the matrix multiplication problem which is faster than the previous one but still suffers from overhead due RAM-to-cache communication. For this particular problem, we report a rather significant difference between the two vectorization options. This is particularly obvious when we observe the difference in terms of MFLOPS: FLOPS are at least twice higher than the former implementation for the matrix multiplication (see FIG. II).

FIG II: a) Floating-point operation per second in million for old and new vectorisation and, b) wall-clock time in ms for an increasing number of material point for an iterative calculation and the old and new vectorisation.

Consequently, we will replace the previous vectorisation by this new technique and make the necessary changes in our revised paper. It also implies to modify results and observations in the computational efficiency section of the paper. However, we will also mention the drop of efficiency due to overheads. In addition, this cost due to RAM-to-cache communication would be an interesting improvement of a future implementation of a vectorised MPM code in MATLAB.

Referee's comment 11

It is well known that MPMs can provide a reasonable global approximation (in terms of force-displacement response) but the computed stress field can be spurious/highly oscillatory due to issues such as locking and/or cell crossing. The authors should present the predicted stress response for the numerical examples presented in the paper and compare them to analytical solutions where available.

Author's reply 11

This is a good suggestion. We will present the stress field for numerical problems in the revised manuscript. Additionally, we will also present the analytical solution of the vertical stress for the elastic column problem and its analytical solution.

Referee's comment 12

The elastic column problem will only have around 2% deformation in terms of the deformed to original height so this problem is not a good test of the convergence of MPMs. The authors refer to the work of Coombs et al. for this problem but they show convergence for the case where the column compresses to around 50% of its initial height. Run the convergence for the case with a Young's modulus of 10kPa, not 1MPa.

Author's reply 12

The referee is totally right; the elastic column will only have a small vertical deformation. We will run the convergence analysis with lower elastic modulus as proposed to achieve a greater deformation of the column.

Referee's comment 13

How have the authors avoided volumetric locking when using isochoric plastic flow? What do the pressure distributions look like through the deformed domains?

Author's reply 13

We did not avoid volumetric locking when using isochoric plastic flow. It is true that any MPM variant suffers from volumetric locking. We should mention this particular problem when treating elasto-plastic problem and we will also add figure to visualize the pressure field for related problems. Additional discussions of flaws due to volumetric locking will be elaborated in the discussion section.

Referee's comment 14

The various comments on the use of cpGIMP and uGIMP are confused and misleading. Just because an analysis is stable it does not mean that it is "appropriate" as the results may be meaningless. The authors cite the work of Coombs to justify the use of the stretch to update the domains, however this technique does not work for problems involving simple shear type deformation, refer to Table 2 of: Coombs, WM, Augarde, CE, Brennan, AJ, Brown, MJ, Charlton, TJ, Knappett, JA, Ghaffari Motlagh, Y & Wang, L (2020). On Lagrangian mechanics and the implicit material point method for large deformation elasto-plasticity. Computer Methods in Applied Mechanics and Engineering 358: 112622. The authors later adopt a determinant of the deformation gradient-type updating method but this has been shown to not converge for simple compression problems as the domains artificially shrink in the non-compressed direction and overlap in the compressed direction (see Figure 8 from the above reference). For example, the statement on lines 76-77 is not correct, or rather it is only correct for some types of problem.

Author’s reply 14

We agree that this various comment might be confused and misleading. As such, we will elaborate the presentation of domain-based MPM and correctly refer to advantages and flaws for each of these domain-based variants. Regarding the determinant of the deformation gradient-type update, the referee comment is absolutely true. However, we selected such approach since it gave the most stable numerical results. But, we will clearly mention the problem of artificial shrink and overlap associated with this variant in the revised paper.

Referee’s comment 15

The speed gains of MILAMIN come from the combination of blocking and vectorisation. Have both techniques been used in this paper? If so, what are the relative speed gains from the different sources? How does the speed gain change with different block sizes?

Author’s reply 15

We did use vectorisation but we did not use blocking. It is true that in MILAMIN the speed gain come from a combination between vectorisation and blocking. However, such technique is applied in assembling local stiffness matrix in the global stiffness matrix, which does not appear in an explicit MPM formulation. Hence, the technique of blocking was disregarder for that concern.

Referee’s comment 16

The authors have not explained by MATLAB is inefficient when working with small amounts of data. This point should be discussed with reference to CPU cache size and RAM-to-cache overheads.

Author’s reply 16

We will discuss this point explicitly when presenting the new vectorisation framework

of the matrix multiplication problem we mentioned previously in our response.

Referee’s comment 17

It would be more appropriate to present the speed gains in terms of flops. There should be a peak in performance if you consider large enough problems.

Author’s reply 17

Thank you for this relevant suggestion. As mentioned previously in our reply, we already started to investigate the speed gain of the matrix-vector operation problem mentioned in Line 198. However, and as mentioned by the referee, we will extend this scope to the computational efficiency section which will greatly benefit from this.

Referee’s comment 18

Figure 2 is misleading as the GIMP domain is fully inside a single element and will have the same connectivity as the standard MP case.

Author’s reply 18

We agree Figure 2 is misleading. Consequently, we will replace the previous Figure 2 by a new one in the revised manuscript (see FIG. III), which we show below

FIG III: Proposed new Figure 2 which shows a correct representation of the connectivity for sMPM and GIMP.

Referee’s comment 19

Line 162 - the authors mention a 30% speed up - what problem/size of problem/number of points, etc?

Author’s reply 19

We mention a 30 % speed up at Line 162 for the calculation of shape function. It is true such statement lacks of evidence and we will provide the reader with a more thoroughly motivated argument. As suggested, we will analyse the performance gain for a variety of number of material points per element and different mesh sizes considering the two kind of vectorization. Thank you for pointing that out.

Referee's comment 20

Where are material points located within the elements when the problems are set up?

Author's reply 20

Regarding the initial location of material points within elements, we preferred a regular distribution of material point within elements. Considering 4 material points per element, their location (in local coordinate) would be [-0.5; 0.5] along x and y direction. We will specify this in the revised paper.

Referee's comment 21

Some key information is missing from the numerical analysis, such as time step sizes and total times which would make it impossible to reproduce the results. This information must be added.

Author's reply 21

We also add information such as time step sizes and total times for the sake of further comparisons, as suggested.

Referee's comment 22

Incomplete sentence on line 359.

Author's reply 22

That is correct, sentence on line 359 will be completed in th

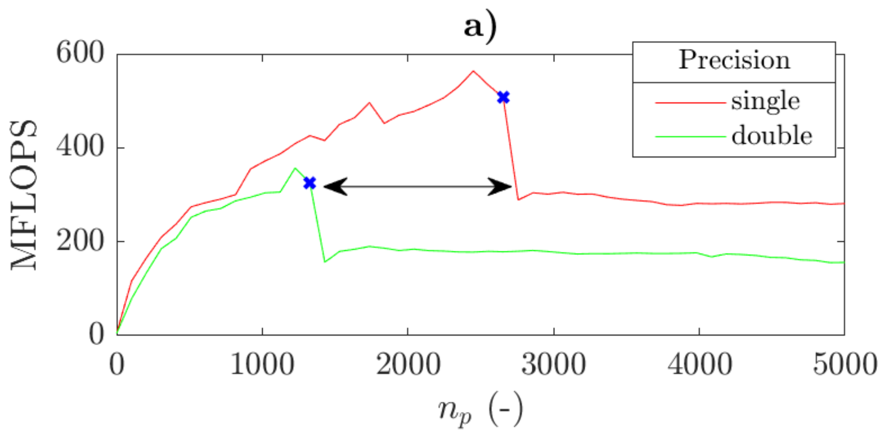

Fig. 1.

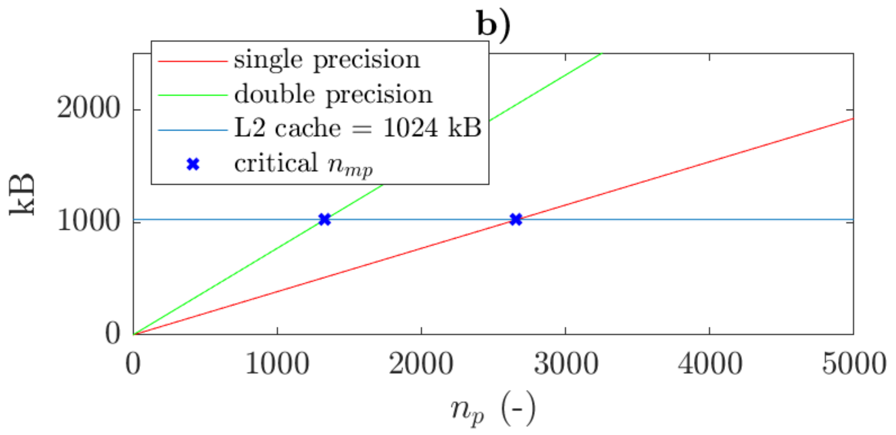
b)
single precision
double precision
L2 cache = 1024 kB
critical n_{mp}
kB
n_p (-)
0
1000
2000
0
1000
2000
3000
4000
5000

Fig. 2.

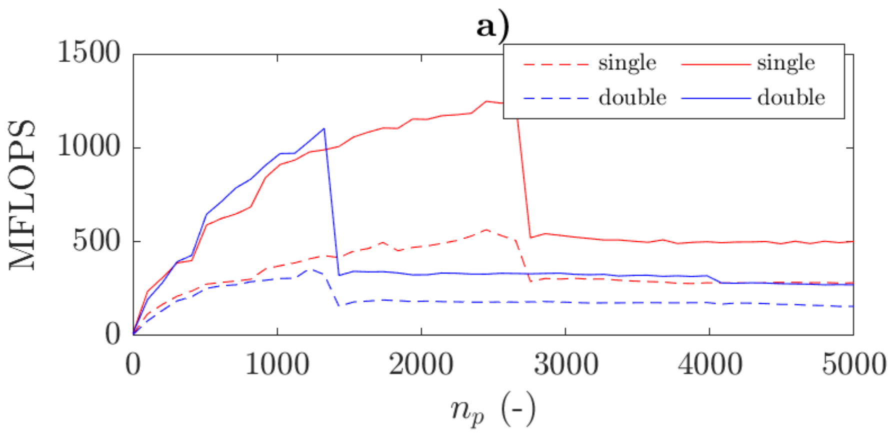
a)
single
double
single
double
MFLOPS
n_p (-)
0
500
1000
1500
0
1000
2000
3000
4000
5000

Fig. 3.

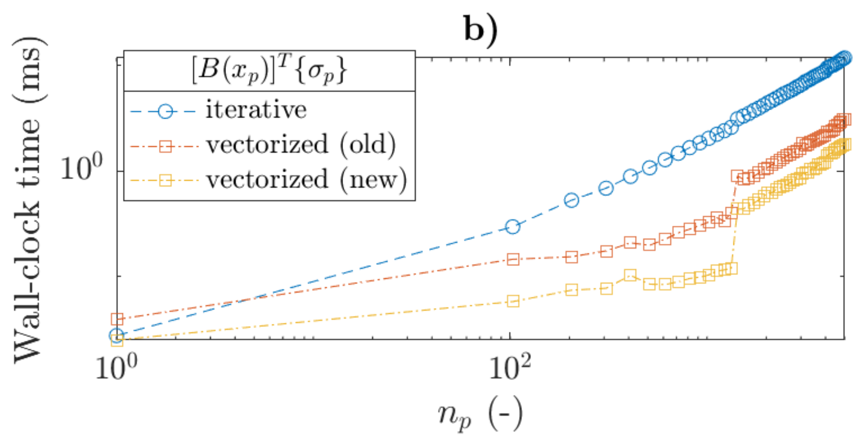
b)
$[B(x_p)]^T\{\sigma_p\}$
iterative
vectorized (old)
vectorized (new)
Wall-clock time (ms)
10^0
10^0
10^2
n_p (-)

Fig. 4.

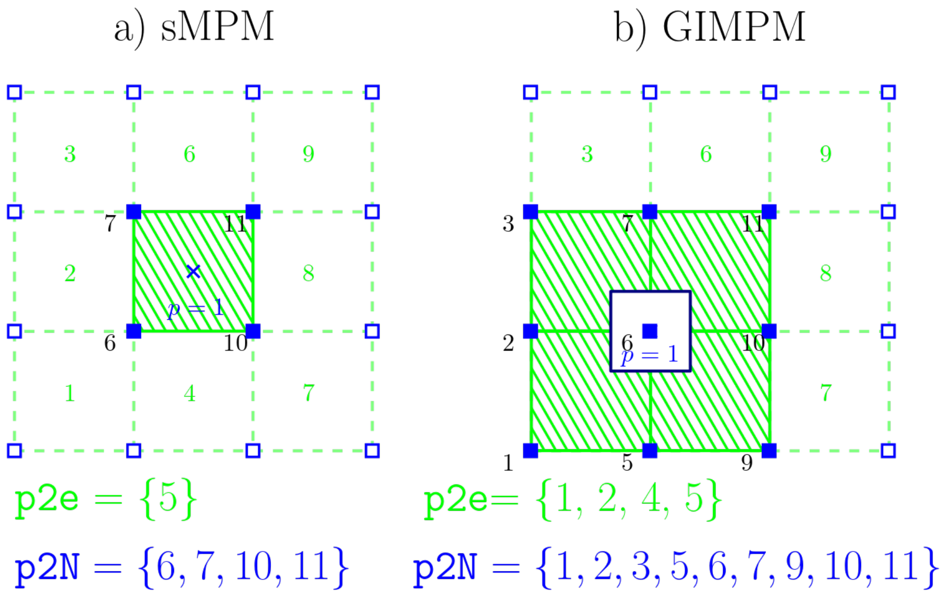
a) sMPM
b) GIMPM
$p = 1$
p2e = {5}
p2N = {6, 7, 10, 11}
p2e= {1, 2, 4, 5}
p2N = {1, 2, 3, 5, 6, 7, 9, 10, 11}

Fig. 5.