The 8th European Congress on Computational Methods in Applied Sciences and Engineering
ECCOMAS Congress 2022
5–-9 June 2022, Oslo, Norway

AN EXTENSION OF A VERY FAST DIRECT FINITE ELEMENT POISSON SOLVER ON LOWER PRECISION ACCELERATOR HARDWARE TOWARDS SEMI-STRUCTURED GRIDS

DUSTIN RUDA[1], STEFAN TUREK[1], DIRK RIBBROCK[1]
AND PETER ZAJAC[1]

[1] Institute for Applied Mathematics and Numerics (LS3)
TU Dortmund University
Vogelpothsweg 87, 44227 Dortmund, Germany
e-mail: dustin.ruda@math.tu-dortmund.de,
stefan.turek@math.tu-dortmund.de,
dirk.ribbrock@math.tu-dortmund.de,
peter.zajac@math.tu-dortmund.de,
www.mathematik.tu-dortmund.de/lsiii/cms/en/lehrstuhl3.html

Key words: Accelerator Hardware, Tensor Core GPUs, NVIDIA A100, Prehandling, Hierarchical Finite Elements, Lower Precision

Abstract. Graphics cards that are equipped with Tensor Core units designed for AI applications, for example the NVIDIA Ampere A100, promise very high peak rates concerning their computing power (156 TFLOP/s in single and 312 TFLOP/s in half precision in the case of the A100). This is only achieved when performing arithmetically intensive operations such as dense matrix multiplications in the aforementioned lower precision, which is an obstacle when trying to use this hardware for solving linear systems arising from PDEs discretized with the finite element method. In previous works, we delivered a proof of concept that the predecessor of the A100, the V100 and its Tensor Cores, can be exploited to a great extent when solving Poisson's equation on the unit square if a hardware-oriented direct solver based on prehandling via hierarchical finite elements and a Schur complement approach is used. In this work, using numerical results on an A100 graphics card, we show that the method also achieves a very high performance if Poisson's equation, which is discretized by linear finite elements, is solved on a more complex domain corresponding to a flow around a square configuration.

1 INTRODUCTION

Previous work has shown how a novel hardware-oriented direct solver for Poisson's equation discretized by finite elements can, under certain conditions, make extensive use of the very high computational power of Tensor Core graphics cards in lower precision and can outperform a standard geometric multigrid solver [1]. So far, results on the unit square were used as a proof of concept and only an outlook on more general domains was given. The goal of this contribution is to show by an example that this method is no less applicable on unstructured coarse grids, which lead to a hierarchy of semi-structured meshes. Initially, we will go into more detail about the hardware used, give an overview of the underlying concept of *prehandling* and outline the derivation of the algorithm. Instead of a very detailed explanation, we refer to relevant existing works at appropriate places.

1.1 The A100 as an example of modern accelerator hardware

As an example of modern accelerator hardware, we consider the NVIDIA A100 SXM4 graphics processing unit (GPU) which includes Tensor Cores. These are special accelerator units that specialize in neural network training and deep learning, and in particular can perform matrix multiplications at very high speed, provided the matrices are dense and single or half precision floating point format is used. The manufacturer specifications of the A100 [2] as well as those of its predecessor V100 [3] and successor, the Hopper H100 [4], whose market launch is expected in the third quarter of 2022, are given in Table 1. The development shows that the trend of lower precision continues to prevail.

Table 1: Specifications of the NVIDIA Volta V100 SXM2, Ampere A100 40GB SXM4 and Hopper H100 SXM graphics cards. The peak performances depending on the precision and availability of Tensor Cores (TC) are given in TFLOP/s.

model	year	memory	double		single		half	
		bandwidth	w/o TC	w/ TC	w/o TC	w/ TC	w/o TC	w/ TC
V100	2017	900 GB/s	7.8	-	15.7	-	31.4	125
A100	2020	1,555 GB/s	9.7	19.5	19.5	156	78	312
H100	2022	3,000 GB/s	30	60	60	1,000	n/a	2,000

The combination of lower precision and Tensor Cores promises a very high performance, but significant utilization of the Tensor Cores seems to be possible only if operations with a high arithmetic intensity (BLAS3), especially dense matrix multiplications, are performed in single or half precision in case of the A100. Our benchmarks show that the peak rates can indeed be reached provided that products of dense matrices are calculated. However, if sparse matrices are used, one falls far short [1].

Of course, it is desirable to exploit Tensor Cores in the context of finite element simulations as, for example, in Computational Fluid Dynamics, but one encounters two major problems: Using single or half precision bears the risk of a deteriorating error, especially if matrices with high condition numbers are involved, and finite element discretizations yield sparse matrices with which Tensor Cores cannot be exploited. Solutions to both problems are presented in the following sections.

1.2 The concept of prehandling

If Poisson's equation, $-\Delta u = f$, in 2D on a domain $\Omega \subset \mathbb{R}^2$ with a homogeneous Dirichlet boundary condition, $u = 0$ on $\partial\Omega$, is discretized on a grid with size h by the finite element method, a stiffness matrix $A \in \mathbb{R}^{N \times N}$ for N unknowns arises with a spectral condition number $\kappa(A) = \mathcal{O}\left(h^{-2}\right)$. This rapidly growing condition number causes a high computational error that

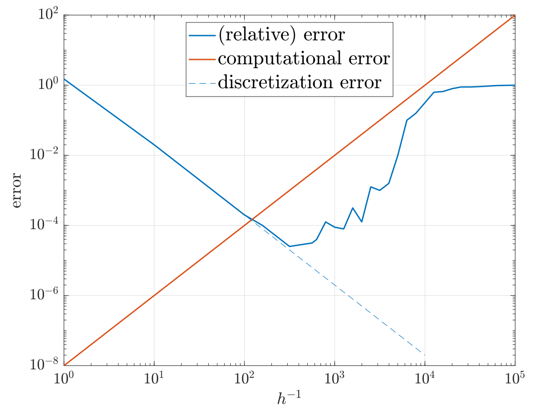


Figure 1: Illustration of the course of the overall, computational and discretization error for Poisson's equation in single precision in 1D. Source: [1]

exceeds the discretization error of $\mathcal{O}\left(h^2\right)$ for (bi-)linear finite elements at a certain refinement level so that the overall error increases. A split of the error – whereby u is the exact solution, u_h is the exact solution to the discrete problem and $\tilde{u}_h$ is the numerical solution to the discrete problem – according to

$$u - \tilde{u}_h = \underbrace{(u - u_h)}_{\text{discr. error}} + \underbrace{(u_h - \tilde{u}_h)}_{\text{comp. error}} \tag{1}$$

clarifies that. More precisely, the computational error is given by $\text{TOL} \cdot \kappa(A)$ with the machine epsilon TOL of the respective precision ($2^{-52} \approx 2.2 \cdot 10^{-16}$ in double, $2^{-23} \approx 1.2 \cdot 10^{-7}$ in single and $2^{-10} \approx 9.8 \cdot 10^{-4}$ in half precision). Equating the approximations for both errors and rearranging, we obtain that the expected turning point of the error is at a grid size of $h \approx \sqrt[4]{\text{TOL}}$. An illustrative example of the course of the three different errors is depicted in Figure 1.

A way to make lower precision accessible while preserving accuracy is decreasing the condition number. As it turned out, classical preconditioning of solvers is not sufficient for this purpose. Instead, we introduced the concept of *prehandling* [5], i.e., transforming a linear system $Ax = b$ into an equivalent form $\tilde{A}\tilde{x} = \tilde{b}$, $x = B\tilde{x}$, whereby the following requirements are met:

- The condition number decreases significantly $\kappa(\tilde{A}) \ll \kappa(A)$,
- the sparsity of the matrix is preserved,
- the transformation is fast.

A candidate that satisfies this in the case of Poisson-like problems in 2D and under minor restrictions also in 3D are *hierarchical finite elements*. It was shown in [6] that in the 2D case the condition number of the stiffness matrix arising from Poisson's equation, or similar elliptic problems, behaves like $\mathcal{O}\left(\left(\log \frac{1}{h}\right)^2\right)$ if given with respect to a hierarchical instead of a nodal basis. In 3D, it is $\mathcal{O}\left(\frac{1}{h}\right)$ [7].

To obtain a hierarchical representation of the stiffness matrix, a hierarchy of grids starting from a coarse grid that is gradually refined is needed. Instead of setting a nodal basis on the fine grid as in the case of standard finite elements, a hierarchical basis is composed of basis functions on all refinement levels. It can be defined inductively as the basis at the coarser level extended by a nodal basis to the newly added nodes. It is not necessary to adapt the assembly of the matrix and right-hand side to a hierarchical basis because, if the linear system is given with respect to a nodal basis as $Ax = b$, the hierarchical formulation is $\tilde{A}\tilde{x} = \tilde{b}$, where $\tilde{A} = S^{\mathsf{T}}AS$, $\tilde{b} = S^{\mathsf{T}}b$, $x = S\tilde{x}$ with a transformation matrix S. In line with the concept of prehandling, the transformed system is to be computed explicitly. The matrix S can be calculated as a product $S = S_j S_{j-1} \dots S_1$, where each factor belongs to a refinement step and corresponds to an interpolation matrix or a square version of a prolongation from the multigrid context (for details, see [1, 5, 6]). The resulting matrix S is a sparse block unit lower-triangular matrix, thus, the costs of the tranformation are low, and the transformed stiffness matrix $\tilde{A} = S^{\mathsf{T}}AS$ is still sparse and has a significantly lower condition number.

A way to further reduce the condition number while only slightly increasing the density of the matrix is additional prehandling by an inverse partial Cholesky factor [6, 8]. To do so, we assume that the matrix $\tilde{A}$ is ordered level-wise, use the entire matrix only on the restriction to the coarsest level, yielding $\tilde{A}^0$, and its diagonal part elsewhere and compute the Cholesky decomposition

$$\begin{pmatrix} \tilde{A}^0 & 0 \\ 0 & \operatorname{diag}\left(\tilde{A}^{1,\dots,j}\right) \end{pmatrix} = LL^{\mathsf{T}}. \tag{2}$$

The new further prehandled matrix is then $L^{-1}\tilde{A}L^{-\mathsf{T}}$ and the right-hand side and solution need to be transformed accordingly. The resulting small condition numbers due to prehandling enable the use of lower precision when solving the system [5].

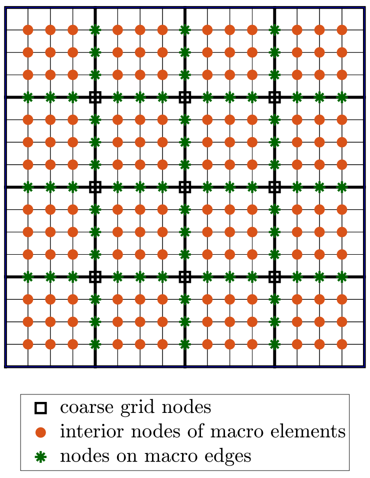
2 A DIRECT SOLVER BASED ON THE HFEM APPROACH

The next goal we pursue, after enabling lower precision through prehandling, is to construct a solver that relies as much as possible on multiplications with dense matrices and thus can exploit the Tensor Cores.

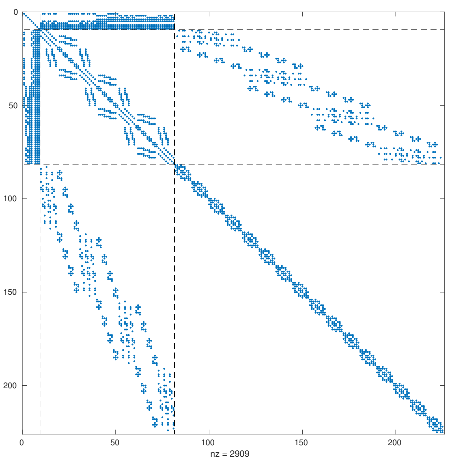
As a starting point, let the linear system arising from Poisson's equation be given after prehandling with hierarchical finite elements and the additional Cholesky approach, now denoted as $Ax = b$ for simplicity. We make a subdivision of the nodes into three different types related to the grid hierarchy and renumber the matrix accordingly. Only the nodes in the interior of the discrete domain are considered because those on the boundary are treated separately due to Dirichlet boundary conditions. The three types are the coarse grid nodes denoted by $\mathcal{C}$, the fine grid nodes on the edges of the coarse grid $\mathcal{E}$ and the nodes in the interior of the coarse grid cells $\mathcal{I}$, numbered cell by cell and geometrically in the same order within each cell. Renumbering the stiffness matrix A in exactly this order provides a special 3×3 block structure. Owing to the Cholesky prehandling on the coarse grid, the upper left block, i.e., the restriction to the coarse grid, is an identity matrix. In certain cases, namely rectangular Q1 and arbitrary P1 grids, there is no coupling between the coarse grid nodes and those in the interior of the macro elements, yielding two zero blocks. Thus, the matrix, reordered in the described manner, can be written in the following block form

$$A = \begin{pmatrix} I & B & 0 \\ B^\mathsf{T} & E & D \\ 0 & D^\mathsf{T} & C \end{pmatrix}. \tag{3}$$

The matrix C, that represents the interaction between the interior nodes of the macro cells, decomposes into independent blocks C_i, as many as there are macro cells, and these blocks are equal if they correspond to similar cells which is an important property regarding the final solver. Two visual examples of the subdivision of the nodes on the unit square and the resulting structure of the prehandled and reordered matrix are given in Figures 2 and 3. Since these are small



(a) Visual subdivision of the nodes. Source: [1]



(b) Sparsity pattern.

Figure 2: (a) Visualization of a Q1 grid and (b) resulting sparsity pattern of the prehandled stiffness matrix on the unit square with 9 coarse grid nodes, 16 square macro cells and 9 nodes within each cell after 2 steps of uniform refinement.

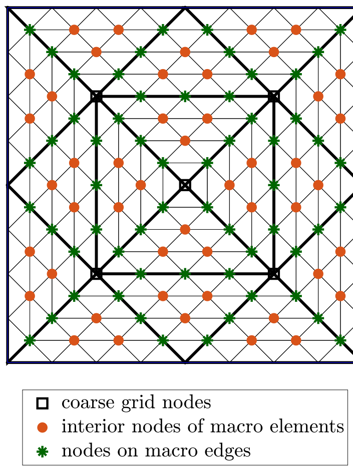
examples that serve the basic understanding, the size ratios of the matrices are distorted. In larger, more application-oriented examples with more refinement steps between the coarse and the fine grid, the matrix C makes up the very largest part of the total matrix A.

To obtain a completely direct solver that consists of many dense matrix multiplications, we also subdivide the right-hand side $b = (b_\mathcal{C}, b_\mathcal{E}, b_\mathcal{I})^\mathsf{T}$ and solution $x = (x_\mathcal{C}, x_\mathcal{E}, x_\mathcal{I})^\mathsf{T}$ and use the structure of the matrix A shown in (3) to apply a Schur complement twice. The resulting three-step algorithm is

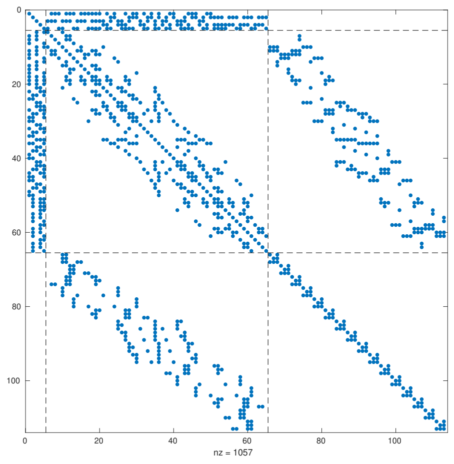
$$x_\mathcal{E} = \Pi^{-1}\left(b_\mathcal{E} - B^\mathsf{T} b_\mathcal{C} - DC^{-1} b_\mathcal{I}\right) \tag{4}$$

$$x_\mathcal{C} = b_\mathcal{C} - Bx_\mathcal{E} \tag{5}$$

$$x_\mathcal{I} = C^{-1}\left(b_\mathcal{I} - D^\mathsf{T} x_\mathcal{E}\right) \tag{6}$$



(a) Visual subdivision of the nodes.



(b) Sparsity pattern.

Figure 3: (a) Visualization of a P1 grid and (b) resulting sparsity pattern of the prehandled stiffness matrix on the unit square with 5 coarse grid nodes, 16 triangular macro cells and 3 nodes within each cell after 2 steps of uniform refinement.

with the auxiliary matrix $\Pi = E - DC^{-1}D^{\mathsf{T}} - B^{\mathsf{T}}B$. It is unconventional to invert matrices explicitly, but this creates high potential for the tensor cores and is not overly expensive because the matrix Π is small – its number of rows equals the number of nodes on the coarse grid edges – and, instead of C, only the submatrices C_i need to be inverted and stored, i.e., one for each group of similar macro cells. If the preprocessing including the computation and inversion of the auxiliary matrices is carried out in double precision, the actual solver consisting of the steps (4) to (6) can be realized in lower precision while preserving accuracy since all involved matrices are well conditioned. The diagonal block structure of C^{-1} can be used to turn the matrix-vector product into a dense matrix-matrix product with the C_i^{-1} if there are many similar macro cells. To sum up, we derived a direct solver for Poisson's quation that includes primarily dense matrix-vector and even matrix-matrix operations in lower precision and is thus well suited to exploit Tensor Core hardware.

The refinement level of the coarse grid is selectable. The finer the coarse grid is, the larger is the matrix Π^{-1} and the smaller are the matrices C_i^{-1}, which affects the complexity of the method. An estimation of the complexity and storage requirement on the unit square discretized with a square grid with Q1 finite elements in [1] yields that, if the coarse grid size is chosen optimally, the direct method requires approximately $12N^{\frac{3}{2}}$ operations for N unknowns. The storage requirement is also $\mathcal{O}\left(N^{\frac{3}{2}}\right)$. A semi-iterative version of the method with a storage cost of $\mathcal{O}(N)$ is under progress.

Multiple right-hand sides

We also consider the case of multiple, $N_{\mathrm{rhs}} \gg 1$, right-hand sides that are solved simultaneously. This corresponds to a matrix as a right-hand side, so the system is $AX = B$ with $X, B \in \mathbb{R}^{N \times N_{\mathrm{rhs}}}$. The solver performs much better in this case because this leads to even more multiplications of dense matrices. The background is that there are novel global-in-time Navier–Stokes solvers that require the simultaneous solution for all time steps to Poisson's equation [9, 10].

3 NUMERICAL RESULTS ON A FLOW AROUND A SQUARE GRID

The presented direct method on the unit square with Q1 finite elements with square elements and its performance on the V100 GPU, also in comparison to a standard multigrid solver on CPU, is extensively examined in [1]. This case is simple in that all macro cells are similar, yielding only one submatrix C_1 that needs to be inverted and used for multiplications with C^{-1}. As a next step, we consider a more complex grid corresponding to a flow around a square problem. The domain with a hole in it is given as $\Omega = (0,4) \times (0,1) \setminus \left[\frac{5}{4}, \frac{7}{4}\right] \times \left[\frac{1}{4}, \frac{3}{4}\right] \subset \mathbb{R}^2$. We define a triangular P1 grid on it as shown in Figures 4 and 5.

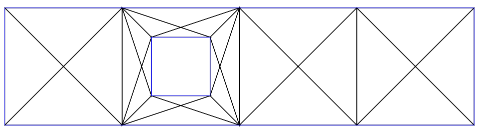


Figure 4: Coarsest grid, Level $L = 0$.

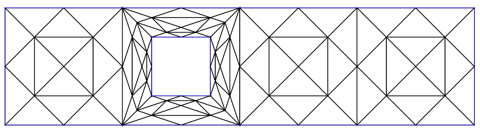


Figure 5: Grid after one step of uniform refinement, Level $L = 1$.

The basic parameters of the grid, i.e., number of unknowns in the interior and number of elements, as well as the density and condition number of the standard finite element stiffness matrix for comparative purposes are listed in Table 2. By taking a closer look at the triangulation, one will notice that it consists of exactly three different types of similar triangles, independently of the refinement level. More precisely, there are isosceles obtuse and scalene triangles, that each make up a share of $2/7$ of all elements, around the hole and isosceles right triangles elsewhere accounting for the remaining $3/7$. There are similar isosceles obtuse triangles of two different sizes, but the corresponding submatrices C_i are nevertheless equal. This leads to the fact that the matrix C consists of only three different submatrices, C_1, C_2 and C_3.

The properties of the matrices, which play a decisive role for the direct method, depending on the fine and coarse grid level, L and L_0, in terms of storage requirement and a resulting estimate of the complexity are listed in Table 3. The density of the matrix D is not listed because its number of nonzero entries relative to N varies only between 0.3 and 2.4. The number of operations of the direct method results mainly from the two multiplications with C^{-1}, which is turned into three dense matrix multiplications, and the multiplication with Π^{-1}. The applications of B and B^{T} as well as D and D^{T} are also taken into account, whereas the cost of vector additions is negligible. The minimum values of the complexity ranging from $10.5N^{\frac{3}{2}}$ to

Table 2: Number of unknowns N, elements, nonzero entries (NNZ) of the standard finite element stiffness matrix relative to N and spectral condition number of this matrix depending on refinement level L on the flow around a square grid.

L	N	#elements	$\frac{\text{NNZ}(A_{\text{FEM}})}{N}$	$\kappa(A_{\text{FEM}})$
0	7	28	1.00	1.88
1	42	112	4.43	10.01
2	196	448	5.53	54.51
3	840	1,792	5.89	257.68
4	3,472	7,168	6.03	1,097.00
5	14,112	28,672	6.09	4,473.84
6	56,896	114,688	6.12	17,994.22
7	228,480	458,752	6.13	72,085.43
8	915,712	1,835,008	6.14	288,459.71

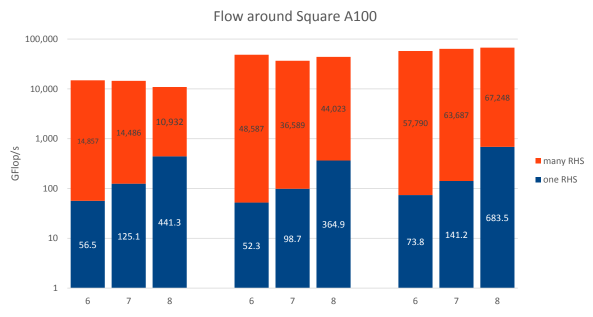
$12N^{\frac{3}{2}}$ are marked in bold. The storage requirement is of course much higher than with standard finite elements, but it can be lowered by choosing a lower coarse grid level, leading to a slightly higher computational demand. A prerequisite for preserving accuracy in single or even half precision are well-conditioned matrices. The results in Table 4 show that this is the case.

Table 3: Number of three types of nodes, nonzero entries of the matrices Π^{-1}, C_i^{-1} (in total for $i = 1, 2, 3$), B and in total (including D) relative to N and total number of operations for the direct method relative to $N^{3/2}$ depending on fine and coarse grid level.

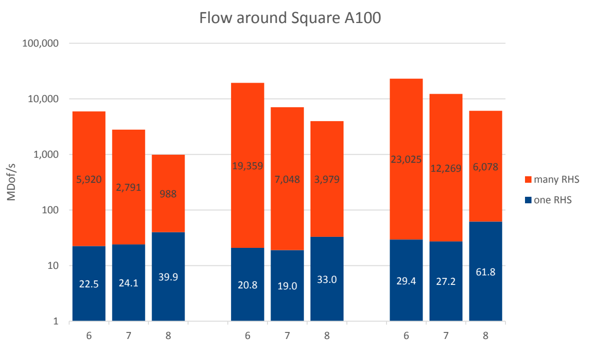
L	L_0	$\lvert\mathcal{C}\rvert$	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{I}\rvert$	$\frac{\text{NNZ}(\Pi^{-1})}{N}$	$\frac{\text{NNZ}(C_{1,2,3}^{-1})}{N}$	$\frac{\text{NNZ}(B)}{N}$	$\frac{\sum \text{NNZ}}{N}$	$\frac{\text{FLOP}}{N^{3/2}}$
6	0	7	2,205	54,684	85.45	201.11	0.05	287.56	32.21
	1	42	4,774	52,080	400.57	11.40	1.45	414.96	**10.55**
	2	196	9,660	47,040	1,640.11	0.58	8.47	1,651.28	15.39
	3	840	18,424	37,632	5,966.04	0.02	31.49	5,999.94	50.82
	4	3,472	31,920	21,504	17,907.87	5e-4	110.21	18,019.85	152.05
7	0	7	4,445	224,028	86.48	840.55	0.02	927.65	66.02
	1	42	9,702	218,736	411.98	50.08	0.74	463.83	17.38
	2	196	19,964	208,320	1,744.90	2.84	4.35	1,753.19	**10.90**
	3	840	39,480	188,160	6,821.91	0.14	16.72	6,840.94	29.43
	4	3,472	74,480	150,528	24,279.02	0.01	63.59	24,345.02	102.25
8	0	7	8,925	906,780	86.99	3,435.98	0.01	3,523.34	134.23
	1	42	19,558	896,112	417.72	209.73	0.37	628.48	33.61
	2	196	40,572	874,944	1,797.60	12.50	2.21	1,813.39	**11.57**
	3	840	81,592	833,280	7,270.03	0.71	8.57	7,280.93	17.01
	4	3,472	159,600	752,640	27,816.78	0.04	33.70	27,852.70	58.65

Table 4: Spectral condition numbers of the matrix Π and the submatrices C_1 (isosceles right), C_2 (scalene) and C_3 (isosceles obtuse triangles).

L	L_0	$\kappa(\Pi)$	$\kappa(C_1)$	$\kappa(C_2)$	$\kappa(C_3)$
6	0	44.25	36.98	52.48	82.26
	1	58.61	23.81	33.02	45.94
	2	45.71	13.57	18.04	23.21
	3	27.96	6.58	8.67	9.44
	4	16.76	2.09	2.40	2.51
7	0	61.05	53.14	76.08	121.93
	1	82.63	36.98	52.48	82.26
	2	68.97	23.81	33.02	45.94
	3	48.92	13.57	18.04	23.21
	4	28.48	6.58	8.67	9.44
8	0	80.83	72.25	104.02	168.20
	1	111.16	53.14	76.08	121.93
	2	97.35	36.98	52.48	82.26
	3	74.08	23.81	33.02	45.94
	4	–	13.57	18.04	23.21



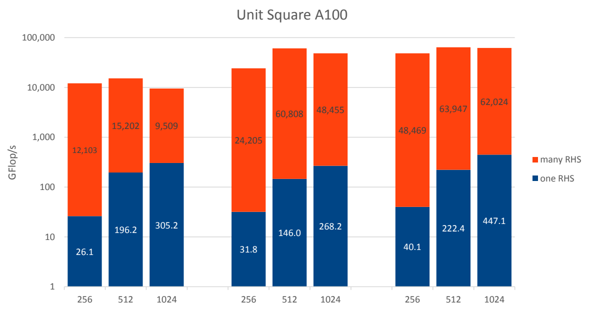
(a)



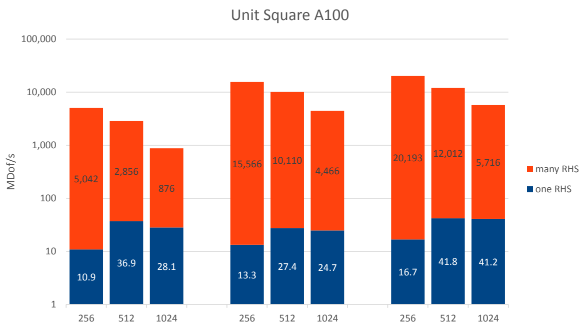
(b)

Figure 6: (a) GFLOP/s and (b) MDof/s for the direct method on the flow around a square grid with one and many right-hand sides depending on the fine grid level on the A100 GPU in double, single and half precision (left, middle and right three columns, respectively).

Following the rather theoretical results, we are now interested in the actual performance of the direct method on the given grid. To this end, the computing times for the solution were measured on an A100 GPU. Since a single graphics card was used, the limit of the refinement level of the fine grid in our tests is 8. Higher levels exceed the storage capacity and would require more units. We conducted tests for three levels for the fine grid with the respective coarse grid level that yields the lowest complexity, thus the pairs of fine and coarse grid level we consider are $(L, L_0) \in \{(6, 1), (7, 2), (8, 2)\}$. Based on the computing times and the known number of operations, we obtain the performance of the overall method in GFLOP/s given in Figure 6a



(a)



(b)

Figure 7: (a) GFLOP/s and (b) MDof/s for the direct method on the unit square (Q1) with one and many right-hand sides depending on h^{-1} where h is the fine grid size on the A100 GPU in double, single and half precision (left, middle and right three columns, respectively).

in double, single and half precision. For the reasons described above, both one and many right sides are used. The highest performance of up to 49 TFLOP/s in single and 67 TFLOP/s in half precision is achieved if many right-hand sides are present. Approximately 74% in double, 31% in single and 22% in half precision of the peak performances of the A100 are reached for many right-hand sides by means of the direct method. Still, there is a vast increase from double to lower precision. As another measure, that also takes the arithmetic expense into account and makes the method comparable to other solvers with a different complexity, we use millions of degrees of freedom solved per second (MDof/s) in Figure 6b. It is consistent with the expectation due to complexity of $\mathcal{O}\left(N^{\frac{3}{2}}\right)$ that the MDof/s values decrease as the problem size increases. The results of several thousand MDof/s are very satisfactory. Tests with a standard geometric multigrid solver in double precision with the software package FEAT3[1] on the unit square discretized by Q1 finite elements on an AMD EPYC 7542 CPU representing standard computers in computer centers carried out on the LiDO3[2] cluster at TU Dortmund University yielded not more than 8 MDof/s, independently of the grid size, for problems with many right-hand sides [1]. Owing to high aspect ratios, the grid used here is unfavorable for iterative solvers such as multigrid methods. Thus, higher iteration numbers are expectable and 8 MDof/s, that are by far outperformed by the direct method, can be seen as an upper bound for a multigrid solver on the flow around a square grid. As a comparison, analogous results for the direct method on the A100 on the unit square are shown in Figure 7. It becomes clear that the results hardly differ depending on the grid, which is a typical behavior of direct solvers, whereas iterative solvers typically deteriorate on more complex grids.

Finally, the accuracy of the method should be addressed, which is significantly dependent on the smoothness of the exact solution function. Here, we define the exact solution

$$u(x,y) = \sin(n\pi x)\sin(n\pi y)\left(x-\frac{5}{4}\right)\left(x-\frac{7}{4}\right)\left(y-\frac{1}{4}\right)\left(y-\frac{3}{4}\right) \tag{7}$$

[1]see `http://www.mathematik.tu-dortmund.de/~featflow/en/software/feat3.html`
[2]see `https://www.lido.tu-dortmund.de/cms/en/home/index.html`

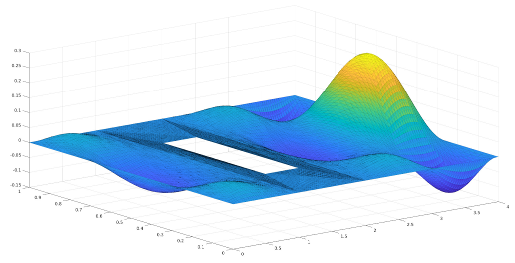


Figure 8: Exact solution, $n = 1$.

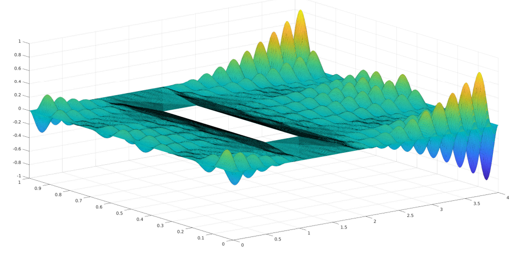


Figure 9: Exact solution, $n = 10$.

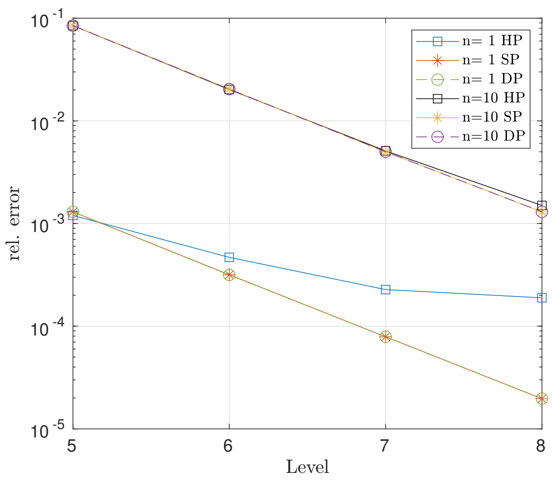


Figure 10: Course of the error obtained with the direct method in half (HP), single (SP) and double precision (DP) as a reference depending on the fine grid level for two values of n.

to Poisson's equation satisfying the homogeneous Dirichlet boundary condition with a parameter $n \in \mathbb{N}$ that controls the smoothness. Graphic representations of a smooth ($n = 1$) and an oscillating ($n = 10$) instance of the function (7) are shown in Figures 8 and 9, respectively. The course of the relative error in the Euclidean norm while refining for both values of n in single and half, compared to double precision, is depicted in Figure 10. There is no difference between the accuracy in single and double precision, even for the very smooth case until level 8. In the case of a more oscillating solution, the accuracy is approximately the same in half precision, but, if the solution is very smooth, the error deviates from that in double and single precision and stagnates at approximately 2×10^{-4}, so still far below 1% which is sufficient for many technical applications. Bearing the machine epsilon of the half precision format in mind, errors lower than that are not to be expected.

4 CONCLUSIONS AND OUTLOOK

In summary, this example shows that the presented direct method works equally well on semi-structured grids and that it is possible to exploit Tensor Cores to a large extent for PDE computing also in this case. However, the direct method is limited to P1 or rectangular Q1 grids and in terms of problem sizes due to the high storage requirement, which is why we are investigating a semi-iterative variant, that includes a small iterative part with respect to a sparse matrix instead of an application of the matrix Π^{-1}. It is slightly less performant but leads to a storage cost of $\mathcal{O}(N)$, is still capable of exploiting Tensor Cores and promises to be more versatile because it is also applicable to finite element spaces of higher order and in 3D.

REFERENCES

[1] Ruda, D., Turek, S., Ribbrock, D. and Zajac, P. Very fast finite element Poisson solvers on lower precision accelerator hardware: A proof of concept study for Nvidia Tesla V100. *The International Journal of High Performance Computing Applications.* (2022)

[2] NVIDIA A100 Tensor Core GPU (40GB SXM) datasheet. Available at: `https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/a100/pdf/nvidia-a100-datasheet-us-nvidia-1758950-r4-web.pdf` (accessed 23 June 2022)

[3] NVIDIA V100 Tensor Core GPU (SXM2) datasheet. Available at: `https://images.nvidia.com/content/technologies/volta/pdf/volta-v100-datasheet-update-us-1165301-r5.pdf` (accessed 23 June 2022)

[4] NVIDIA H100 Tensor Core GPU (SXM) datasheet. Available at: `https://resources.nvidia.com/en-us-tensor-core/nvidia-h100-datasheet` (accessed 23 June 2022)

[5] Ruda, D., Turek, S., Zajac, P. and Ribbrock, D. The Concept of Prehandling as Direct Preconditioning for Poisson-Like Problems. *Numerical Mathematics and Advanced Applications ENUMATH 2019. Lecture Notes in Computational Science and Engineering.* Vermolen, F.J., Vuik, C. (eds). (2021) **139**:1011–1019.

[6] Yserentant, H. On the Multi-Level Splitting of Finite Element Spaces. *Numerische Mathematik.* (1986) **49**:379–412.

[7] Ong, M.E.G. Hierarchical Basis Preconditioners in Three Dimensions. *SIAM Journal on Scientific Computing.* (1997) **18(2)**:479–498.

[8] Deuflhard, P., Leinen, P. and Yserentant, H. Concepts of an adaptive hierarchical finite element code. *IMPACT of Computing in Science and Engineering.* (1989) **1(1)**: 3–35.

[9] Dünnebacke, J., Turek, S., Lohmann, C., Sokolov, A. and Zajac, P. Increased space-parallelism via time-simultaneous Newton-multigrid methods for nonstationary nonlinear PDE problems. *The International Journal of High Performance Computing Applications.* (2021) **35(3)**:211–225.

[10] Lohmann, C. and Turek, S. On the design of global-in-time Newton-Multigrid-Pressure Schur complement solvers for incompressible flow problems. To appear in: *Journal of Mathematical Fluid Mechanics.* (in progress)