technische universität dortmund

fakultät für mathematik

Very Fast FEM Poisson Solvers on Lower Precision Accelerator Hardware

Dustin Ruda, Stefan Turek, Dirk Ribbrock, Peter Zajac

Institute for Applied Mathematics (LS 3), TU Dortmund University

ECCOMAS Congress 2022, 5–9 June 2022, Oslo, Norway

8 June 2022

Motivation

Prec.	double	double + TC	single	single + TC	half	half + TC
V100	7.8	-	15.7	-	31.4	**125**
A100	9.7	19.5	19.5	**156**	-	**312**
H100	30	60	60	500	120	1,000

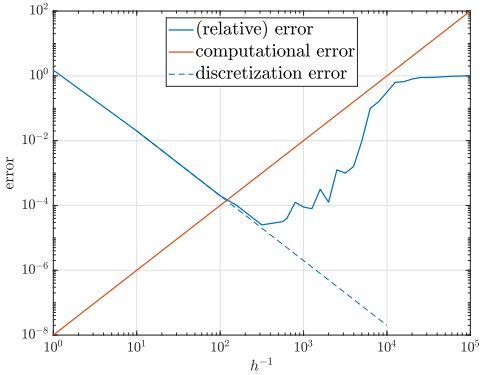
TFlop/s peak rates for **NVIDIA V100** (2017), **A100** (2020) and **H100** (Q3 2022)
(similar: AMD Matrix Core)

- 100+ TFlop/s only achievable in **lower precision** by **Tensor Cores** (TC)
- Peak rates only achievable with **dense** matrix operations
- **Aim**: Profitable use of this hardware for linear systems in FE simulations (CFD)
- Consider **Poisson's equation**: Global-in-time Navier–Stokes solver allows for solving Pressure Poisson problems for all time steps at once → many right hand sides (RHS)

↯ Standard FEM solvers (MG) require double precision (DP) and include large, sparse matrices → *Prehandling* and new *Schur complement-based methods*

How to handle ill-conditioned Poisson-like Problems

- **Split the error:**
  $u - \widetilde{u}_h = (u - u_h) + (u_h - \widetilde{u}_h)$
- **Discr. Error:**
  $$\|u - u_h\| = \mathcal{O}\left(h^{p+1}\right)$$
  - depending on **FEM space** and **smoothness**
  - Here for simplicity: $p = 1$
- **Comp. Error:**
  $\|u_h - \widetilde{u}_h\| \approx \mathrm{cond} \cdot$ "data error"
  - Data error at least TOL of respective precision
  - Poisson: $\mathrm{cond}\left(A_h\right) = \mathcal{O}\left(h^{-2}\right)$

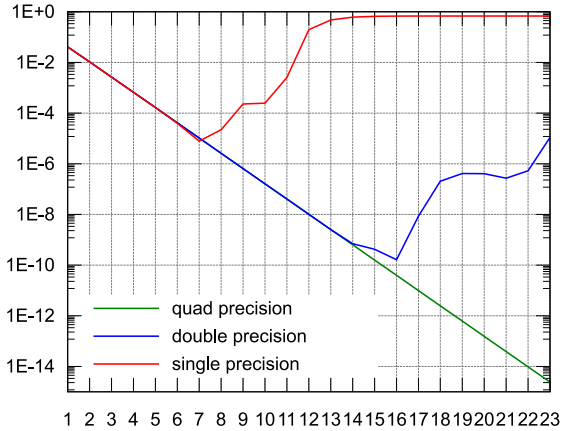


Source: Ruda et al, 2022

- Critical h at intersection of both errors: Discr. Error $\approx$ Comp. Error $\Rightarrow h \approx (\mathrm{cond} \cdot \mathrm{TOL})^{\frac{1}{2}}$

How to handle ill-conditioned Poisson-like Problems

- Critical grid width: $h \approx (\mathrm{cond} \cdot \mathrm{TOL})^{\frac{1}{2}}$
- Poisson $-\Delta u = f$: Substitute $\mathrm{cond} = \mathcal{O}\left(h^{-2}\right) \Rightarrow h \approx \mathrm{TOL}^{\frac{1}{4}}$
- 1D example: $(\mathrm{TOL}_{\mathrm{SP}})^{\frac{1}{4}} = 2^{-5.75}$
  $(\mathrm{TOL}_{\mathrm{DP}})^{\frac{1}{4}} = 2^{-13}$
- Wish: $\mathrm{cond} = \mathcal{O}(1) \Rightarrow h \approx \mathrm{TOL}^{\frac{1}{2}}$
  $\rightarrow$ SP (and even HP?) possible



L^2-error with standard FEM in 1D, $h = 2^{-\mathrm{level}}$
Source: Ruda et al., 2022

Concept of Prehandling of Linear Systems

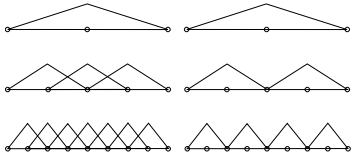
Preconditioning: $x^{k+1} = x^k - C^{-1}\left(Ax^k - b\right)$

Prehandling: $x^{k+1} = x^k - \left(C^{-1}Ax^k - C^{-1}b\right) = x^k - \left(\widetilde{A}x^k - \widetilde{b}\right)$

- Yields same solution (if converged and with "infinite precision") and same iteration numbers, but $\mathrm{cond}(A) \lesseqgtr \mathrm{cond}\left(\widetilde{A}\right)$ since different linear systems
- **Central idea:** Explicitly transforming $Ax = b$ into equivalent $\widetilde{A}\widetilde{x} = \widetilde{b}$, $B\widetilde{x} = x$ with:
  1) $\mathrm{cond}\left(\widetilde{A}\right) \ll \mathrm{cond}(A)$
  2) $\widetilde{A}$ only moderately less sparse than A
  3) Tranformation to $\widetilde{A}$, $\widetilde{b}$ (resp. x via B) fast (i.e. $\mathcal{O}(N \log N)$)

HFEM: Ideas, Realization & Properties

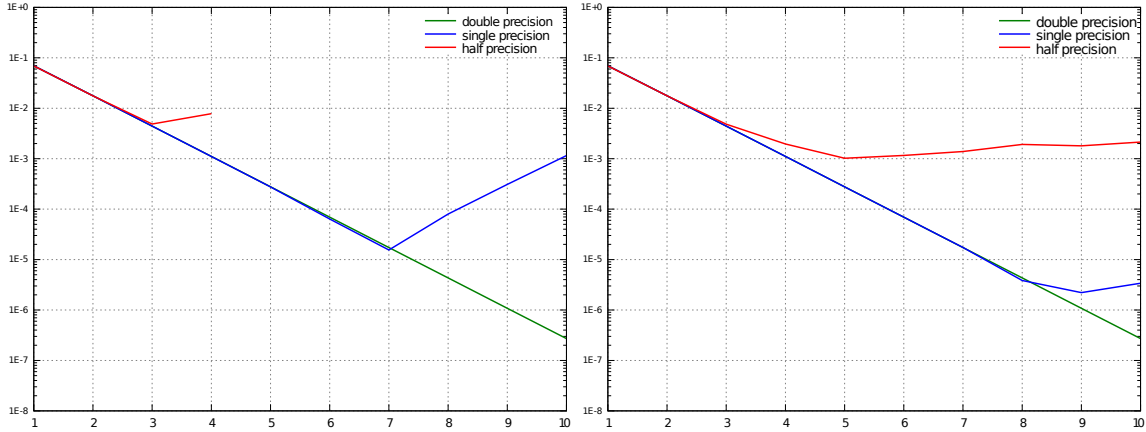
- Only candidate for prehandling so far: HFEM
- **Idea:** Use of **hierarchical** instead of **nodal basis** starting from a coarse grid
- Transform linear system $\widetilde{A} = S^{\mathsf{T}} A S$, $\widetilde{b} = S^{\mathsf{T}} b$, $x = S\widetilde{x}$



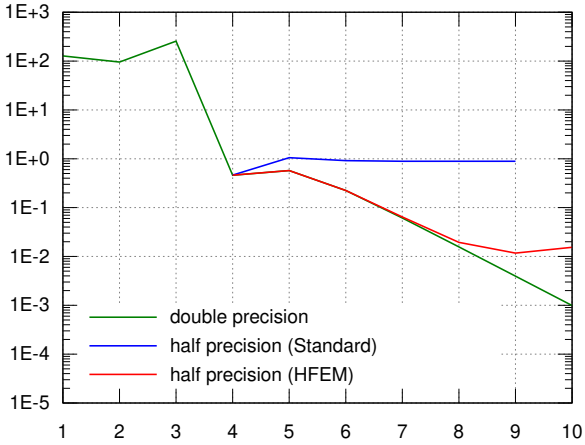
nodal (l) and hier. (r) basis
Source: Deuflhard et al., 1989

- $\operatorname{cond}\left(\widetilde{A}\right) = \mathcal{O}\left(\left(\log \frac{1}{h}\right)^2\right)$ in 1D, 2D; FEM: $\operatorname{cond}(A) = \mathcal{O}\left(\left(\frac{1}{h}\right)^2\right)$
- Add. partial Cholesky prehandling: $\begin{pmatrix} \widetilde{A}_0 & 0 \\ 0 & \widetilde{D} \end{pmatrix} = L^{\mathsf{T}} L \to L^{-1} \widetilde{A} L^{-\mathsf{T}}$
- Remark: in 3D $\operatorname{cond}\left(\widetilde{A}\right) = \mathcal{O}\left(\frac{1}{h} \log \frac{1}{h}\right)$ resp. $\mathcal{O}\left(\frac{1}{h}\right) \to$ Possible in SP

HFEM: Numerical results (errors)



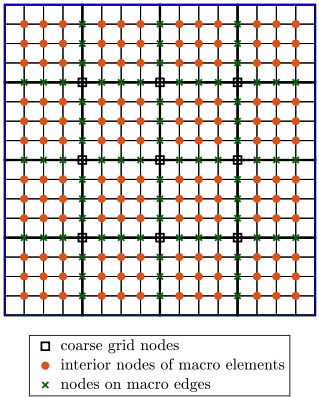
L^2-errors for different levels in 2D in DP, SP, HP without (left) and with (right) **prehandling via HFEM** for **"smooth"** solution. Source: Ruda et al., 2020



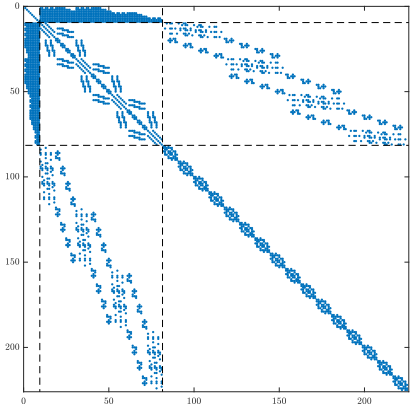
L^2-errors for different levels in 2D in DP, SP, HP without and with **prehandling via HFEM** for **strongly oscillating** solution
Source: Ruda et al., 2022

- Fine h for tolerance of $\approx 1\%$ for complex problems
  $\rightarrow$ large problems
  $\rightarrow$ requires HPC

- **Objective:** Construct solver consisting as much as possible on multiplications with **dense**, **well-conditioned matrices**
- **Starting Point:** Linear system after prehandling via **HFEM+Cholesky** $Ax = b$
- Subdivide nodes into 3 types ($\mathcal{C}$, $\mathcal{E}$, $\mathcal{I}$) and renumber A accordingly



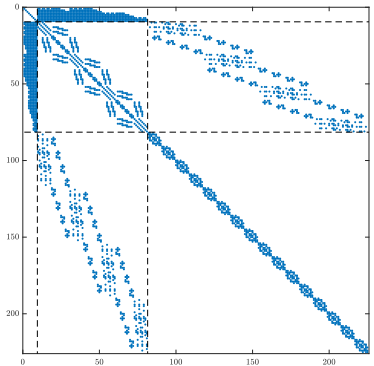
Source: Ruda et al., 2022



Source: Ruda et al., 2022

$$\begin{pmatrix} I & B & 0 \\ B^\mathsf{T} & E & D \\ 0 & D^\mathsf{T} & C \end{pmatrix} \begin{pmatrix} x_\mathcal{C} \\ x_\mathcal{E} \\ x_\mathcal{I} \end{pmatrix} = \begin{pmatrix} b_\mathcal{C} \\ b_\mathcal{E} \\ b_\mathcal{I} \end{pmatrix}$$

- D, E are sparse
- B is moderately dense
- C decomposes into **independent** blocks (as many as macro cells)



Source: Ruda et al., 2022

- **Blocks C_i of C are equal if they correspond to similar macro cells**
- **Only C grows like N** ($= \#\text{Dof}$)
- Applying **Schur complement** $\rightarrow$ semi-iterative method
- Applying further Schur complement $\rightarrow$ completely direct method

Semi-iterative Method

$\Lambda = E - DC^{-1}D^{\mathsf{T}}$

Use conjugate gradient method to solve

$$\underbrace{\begin{pmatrix} I & B \\ B^{\mathsf{T}} & \Lambda \end{pmatrix}}_{\Sigma} \begin{pmatrix} x_{\mathcal{C}} \\ x_{\mathcal{E}} \end{pmatrix} = \begin{pmatrix} b_{\mathcal{C}} \\ b_{\mathcal{E}} - DC^{-1}b_{\mathcal{I}} \end{pmatrix}$$

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

Direct Method

$\Lambda = E - DC^{-1}D^{\mathsf{T}}$

$\Pi = \Lambda - B^{\mathsf{T}}B$

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

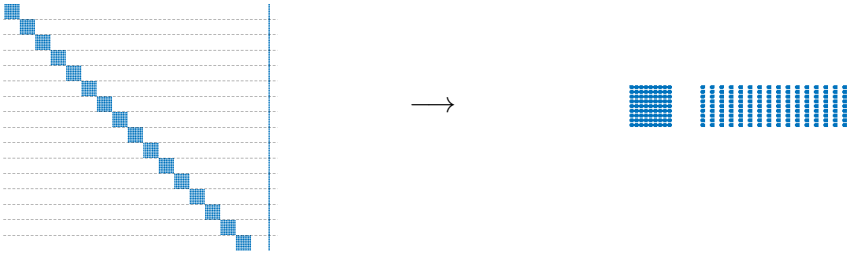
$$x_{\mathcal{C}} = b_{\mathcal{C}} - Bx_{\mathcal{E}}$$

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

- Matrices Σ, Λ, Π, C well-cond. (5–50 on unit square with Q1)
- Block structure of C: **Only C_i^{-1} computed and stored**
- Semi-iterative: Less storage consuming
- Direct: More storage consuming but even higher potential for TC, especially in case of many RHS

Multiplication with C^{-1}

- Both methods require 2 multiplications with C^{-1}
- Efficient implementation by transforming into dense matrix product (also if $\#\text{RHS} = 1$):



- Complexity $\mathcal{O}\left(N^{\frac{3}{2}}\right)$ but fast calculation by TC

Storage and Computational Cost of the Direct Method

- Consider equidistantly refined unit square, Q1
- Let $h_0 = 2^\ell \sqrt{h}$, $\ell = \ldots, -1, 0, 1, 2, \ldots$
- Relevant for **storage**: Π^{-1}, C_i^{-1}
- Relevant for **FLOP**: Π^{-1}, $C^{-1}(2\times)$

Direct Method

$\Lambda = E - DC^{-1}D^\mathsf{T}$, $\Pi = \Lambda - B^\mathsf{T}B$

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

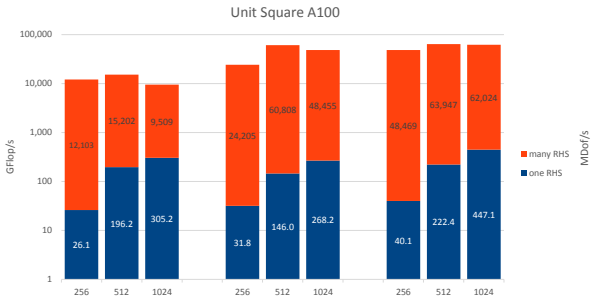
$h = \frac{1}{1024}$:

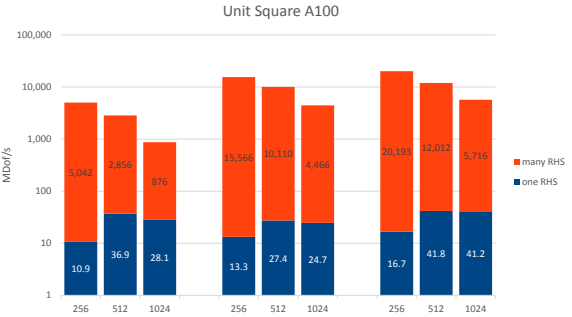
ℓ	-1	0	1	2	3
Total NNZ/N	16,400	14,100	1,000	500	4,200

ℓ	-1	0	1	2	3
#FLOP/$N^{3/2}$	33	**12**	18	65	256

- Best choice in terms of complexity: $h_0 = \sqrt{h}$ (or $\ell = 1$ considering storage)
- $\approx 12N^{3/2} \to$ 12,000 (SC) vs. 1,000 FLOP (MG) for $h = \frac{1}{1024}$

Direct Method: Unit Square on A100



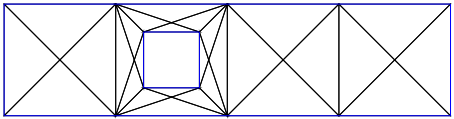


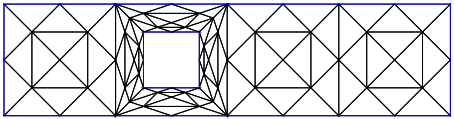
GFLOP/s (left) and **MDof/s** (right) with direct method on **A100** with one and many RHS depending on h^{-1} in **DP**, **SP** and **HP** (left, middle and right 3 columns, respectively)

→ **Up to 60 TFlop/s (for problems with many RHS)**

→ More arithmetic work ($\times 12$), but still much faster than standard MG solver on x64 AMD CPU (8 MDof/s for many rhs)

Unstructured Coarse Grids

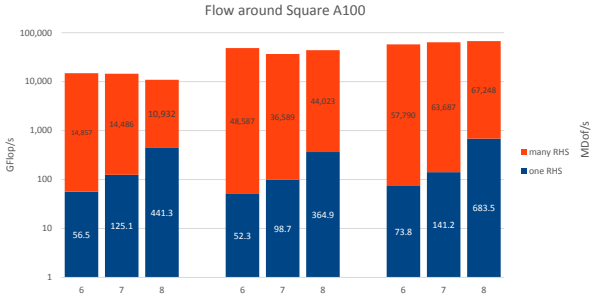


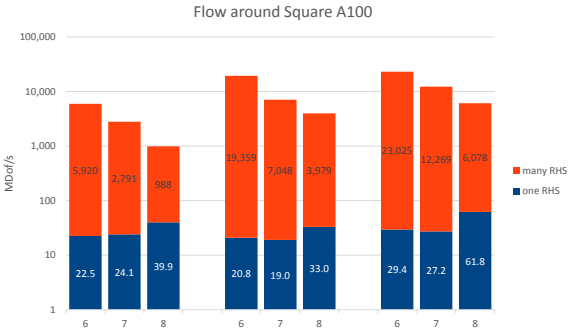


- So far: Analysis and numerical tests on unit square
- **Direct method** also applicable to "arbitrary" P1 grids
- Coarse grids with many similar cells are advantageous $\rightarrow$ few different C_i
- Example "flow around a square"
  - 3 Groups of similar cells $\rightarrow$ $C_1^{-1}, C_2^{-1}, C_3^{-1}$ must be calculated and stored

L (L_0) N	$\frac{\#\text{FLOP}}{N^{\frac{3}{2}}}$	$\frac{\text{NNZ}}{N}$
6 (1) 56,896	10.5	410
7 (2) 228,480	10.9	1,750
8 (2) 915,712	11.6	1,810

Direct Method: Flow around a Square on A100





GFLOP/s (left) and **MDof/s** (right) with direct method on **A100** with one and many RHS depending on h^{-1} in **DP**, **SP** and **HP** (left, middle and right 3 columns, respectively)

Limitations of the Direct Method

- High storage requirement of $\mathcal{O}\left(N^{\frac{3}{2}}\right)$ due to Π^{-1}
- Limit of fine grid width in our tests: $h = \frac{1}{1024}$ (on one GPU)
- Hardly applicable to 3D because storage requirement of $\mathcal{O}\left(N^{\frac{5}{3}}\right)$
- Requirement for simple form of the direct method: No coupling between nodes in $\mathcal{C}$ and $\mathcal{I}$ (coarse grid and interior nodes)
  - Rectangular Q1 grids
  - "arbitrary" P1 grids
- **Less memory consuming, more versatile but also less performant variant: Semi-iterative method**

Semi-iterative Method: Storage Requirement

$\frac{1}{h}$	$\frac{N}{10^6}$	$\frac{1}{h_0}$	Σ	C_i^{-1}	D	total
1024	1.05	16	15	15.1	1.0	**31**
		32	25	0.9	1.6	**27**
2048	4.19	32	19	3.8	1.0	**24**
		64	40	0.2	1.6	**42**
4096	16.77	32	16	15.5	0.7	**32**
		64	27	0.9	1.0	**29**

Number of nonzero entries relative to N

Semi-iterative Method

$\Lambda = E - DC^{-1}D^{\mathsf{T}}$
Use conjugate gradient method to solve

$$\underbrace{\begin{pmatrix} I & B \\ B^{\mathsf{T}} & \Lambda \end{pmatrix}}_{\Sigma} \begin{pmatrix} x_{\mathcal{C}} \\ x_{\mathcal{E}} \end{pmatrix} = \begin{pmatrix} b_{\mathcal{C}} \\ b_{\mathcal{E}} - DC^{-1}b_{\mathcal{I}} \end{pmatrix}$$

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

- Storage requirement: $30N - 40N$ in SP (in comparison: $9N$ with standard FEM in DP)

Semi-iterative Method: Performance Estimate

- **Basic composition of the method**:
  - $1 \times D$ and $1 \times C^{-1}$ to compute RHS
  - **Iterative step**: $1 \times \Sigma$, 3 AXPY and 2 dot products per iteration
    $\rightarrow \#\text{iter}\,[2\mathrm{NNZ}(\Sigma) + 6\#\text{rows} + 4\#\text{rows}]$,
    $\#\text{rows} = \mathcal{O}\left(N^{\frac{3}{4}}\right)$
  - Intermediate step: $1 \times D^{\mathsf{T}}$
  - **Direct step**: $1 \times C^{-1}$

Semi-iterative Method

$\Lambda = E - DC^{-1}D^{\mathsf{T}}$
Use conjugate gradient method to solve

$$\underbrace{\begin{pmatrix} I & B \\ B^{\mathsf{T}} & \Lambda \end{pmatrix}}_{\Sigma} \begin{pmatrix} x_{\mathcal{C}} \\ x_{\mathcal{E}} \end{pmatrix} = \begin{pmatrix} b_{\mathcal{C}} \\ b_{\mathcal{E}} - DC^{-1}b_{\mathcal{I}} \end{pmatrix}$$

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

- Entire method in SP on A100
- Majority of the work: Dense matrix operations; Small part: sparse $\times$ dense and BLAS1

Semi-iterative Method: Performance Estimate

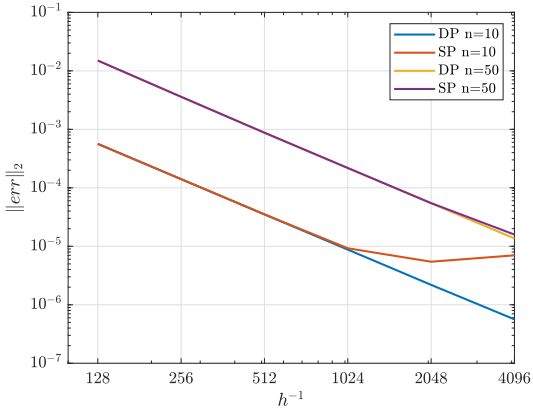
$\Sigma *$ dense	AXPY	dot product	dense $*$ dense	$D *$ dense
2,375	227	321	**150,000**	1,200

GFlop/s in benchmarks on A100 in SP

$\frac{1}{h}$	$\frac{1}{h_0}$	#iter	FLOP/N it.	FLOP/N dir.	time it.+dir.	GFLOP/s	MDof/s
1024	16	30	913	15,400	0.43 + 0.11	31,445	**1,926**
	32	24	1,217	3,615	0.60 + 0.03	8,206	1,698
2048	32	28	1,085	15,400	2.04 + 0.43	27,919	**1,694**
	64	23	1,881	3,611	3.53 + 0.10	6,333	1,153
4096	32	31	1,011	63,543	7.43 + 7.10	74,476	1,154
	64	25	1,374	15,391	10.17 + 1.72	23,636	**1,410**

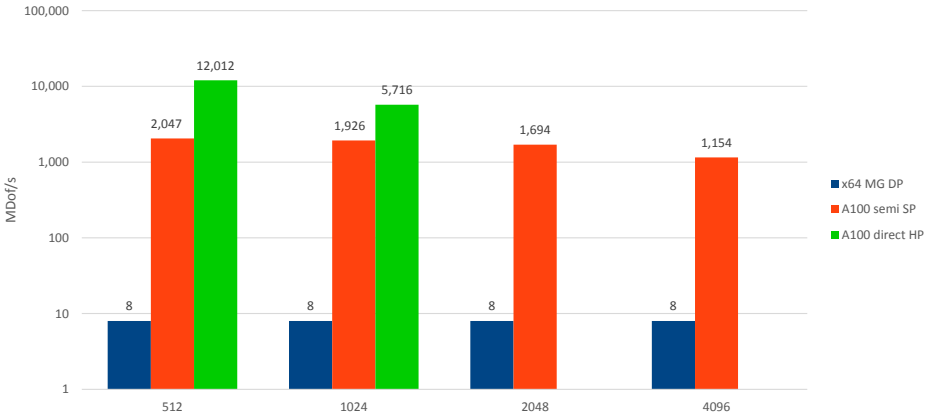
Performance estimate

Semi-iterative Method: Accuracy



Relative error with semi-iterative method in DP and SP for differently smooth solutions

Comparison: MG – semi-iterative – direct



Comparison of **MDof/s** for many RHS with **MG** in **DP** on **AMD CPU**, **direct** method in **HP** on **A100** and **semi-iterative** in **SP** on **A100** (estimate)

Outlook and Conclusion

- Implementation of the semi-iterative method on GPU
- Prehandling in 3D
- Analysis of suitable preconditioners for the iterative step and initial guesses for the solution vector to reduce number of iterations
- Testing semi-iterative Method for other FE spaces and in 3D
- Implementation into FEATFLOW software

- **Conclusion: It is possible to exploit Lower-Precision Accelerator Hardware for PDE computing** (under certain conditions)

References

- Yserentant, H. (1986). On the Multi-Level Splitting of Finite Element Spaces. Numer. Math., Vol. 49, pp. 379–412, DOI: 10.1007/BF01389538
- Deuflhard, P., Leinen, P., Yserentant, H. (1989). Concepts of an adaptive hierarchical finite element code. IMPACT of Computing in Science and Engineering, Vol. 1, Issue 1, pp. 3–35, DOI: 10.1016/0899-8248(89)90018-9
- Ruda, D., Turek, S., Zajac, P. & Ribbrock, D. (2020). The Concept of Prehandling as Direct Preconditioning for Poisson-like Problems. Vermolen, F., Vuik, C., Lecture Notes in Computational Science and Engineering, 139, 1011-1019, Numerical Mathematics and Advanced Applications Enumath 2019, Springer, 2020, DOI: 10.1007/978-3-030-55874-1_100
- Ruda, D., Turek, S., Zajac, P. & Ribbrock, D. (2022). Very fast finite element Poisson solvers on lower precision accelerator hardware: A proof of concept study for Nvidia Tesla V100, International Journal of High Performance Computing Applications, 2022, DOI: 10.1177/10943420221084657