Accurate and Reproducible Conjugate Gradient in Hybrid Parallel Environments

Roman Iakymchuk, Daichi Mukunoki, Takeshi Ogita, Katsuhisa Ozaki, Stef Graillat

Accurate and Reproducible Conjugate Gradient in Hybrid Parallel Environments

Roman Iakymchuk[1,2], Daichi Mukunoki[3], Takeshi Ogita[4], Katsuhisa Ozaki[5], and Stef Graillat[1]

[1]Sorbonne Université, LIP6, France [2]Fraunhofer ITWM, Germany [3]RIKEN Center for Computational Science, Japan [4]Tokyo Woman's Christian University, Japan [5]Shibaura Institute of Technology, Japan

Introduction

The Preconditioned Conjugate Gradient (PCG) method is often used for solving linear systems of equations arising in various numerical simulations. While being widely used, the solver is also known for its lack of accuracy while computing the residual. Additionally, implementations of the same PCG algorithm, e.g. with Jacobi preconditioner, in different parallel environments may lead to different results as the table shows for the matrix of a 3D Poisson's equation with 27 stencil points, n=4,019,679, and $cond(A) = 10^{12}$.

Algorithm 1 CG solving $\boldsymbol{Ax} = \boldsymbol{b}$

1: $\boldsymbol{p}_0 = \boldsymbol{r}_0 = \boldsymbol{b} - \boldsymbol{Ax}_0$ // SpMV
2: $\rho_0 = \boldsymbol{r}_0{}^T \boldsymbol{r}_0$ // DOT
3: $i = 0$
4: **while** 1 **do**
5: $\quad \boldsymbol{q}_i = \boldsymbol{Ap}_i$ // SpMV
6: $\quad \alpha_i = \rho_i / \boldsymbol{p}_i{}^T \boldsymbol{q}_i$ // DOT
7: $\quad \boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \alpha_i \boldsymbol{p}_i$ // AXPY
8: $\quad \boldsymbol{r}_{i+1} = \boldsymbol{r}_i - \alpha_i \boldsymbol{q}_i$ // AXPY
9: $\quad$ **if** $||\boldsymbol{r}_{i+1}||/||\boldsymbol{b}|| < \epsilon$ **then** // NRM2
10: $\quad\quad$ **break**
11: $\quad$ **end if**
12: $\quad \rho_{i+1} = \boldsymbol{r}_{i+1}{}^T \boldsymbol{r}_{i+1}$ // DOT
13: $\quad \beta_i = \rho_{i+1} / \rho_i$
14: $\quad \rho_i = \rho_{i+1}$
15: $\quad \boldsymbol{p}_{i+1} = \boldsymbol{r}_{i+1} + \beta_i \boldsymbol{p}_i$ // SCAL
16: $\quad i = i + 1$
17: **end while**

Iter	Absolute Residual Norm ($\|\|\boldsymbol{r}_i\|\|$)	
	MPI 1 process	*MPI 48 processes*
0	0x1.19f179eb7f033p+49	0x1.19f179eb7f033p+49
2	0x1.f86089ece**5bd4**p+38	0x1.f86089ece**af76**p+38
9	0x1.fc59a29d3**599a**p+28	0x1.fc59a29d3**2d1b**p+28
10	0x1.74f5ccc**1d03cb**p+22	0x1.74f5ccc**201246**p+22
...	...	...
40	0x1.7031058**dd6bcf**p-19	0x1.7031058**eaf4c2**p-19
42	0x1.4828f76**d1aa3**p-23	0x1.4828f76**bda71a**p-23
45	0x1.8646260a**2dae8**p-26	0x1.8646260a**6da06**p-26
47	0x1.13fa97e**1e76bf**p-33	0x1.13fa97e**240f7c**p-33

The same trend holds for the SuiteSparse matrices also in terms of iterations, e.g. for the gyro_k matrix on the MN4 cluster at BSC and Tintorrum (TR) at UJI, both in Spain [2].

Matrix	$cond(A)$	ϵ	MPI@MN4	MPI+OMP@MN4	MPI@TR	MPI+OMP@TR
gyro_k	$1.10e+09$	10^{-8}	16,557	16,064	16,518	16,623

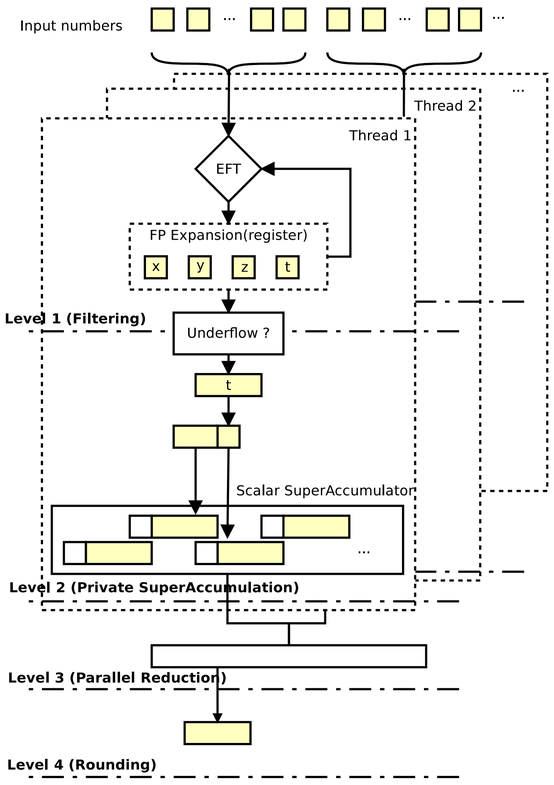
Our goal is to secure the accuracy and reproducibility of PCG in hybrid parallel environments with the minimal possible overhead.

Strategies for Accurate and Reproducible (P)CG

Our strategies are reinforced with programmability components such as the explicit use of *fma* instructions and a careful re-arrangement of computations. The reproducibility of (P)CG is guaranteed via reproducibility of its building blocks: *dot product, axpy, spmv.*

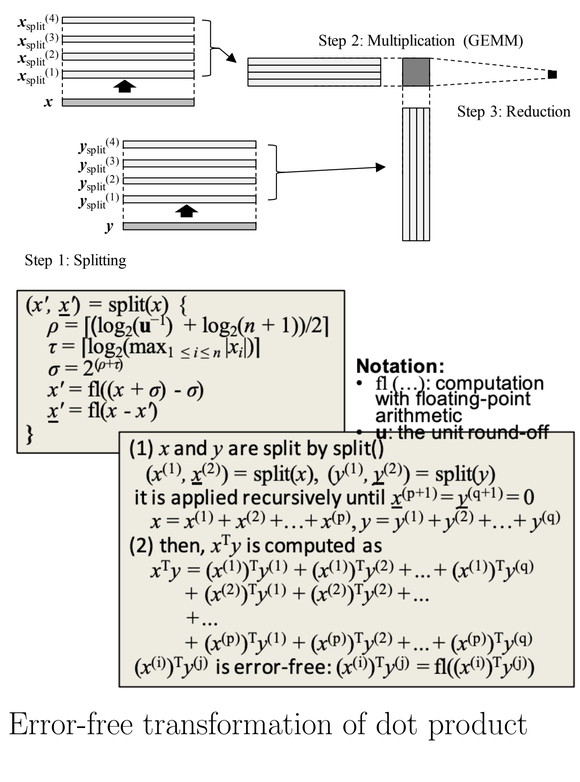
ExBLAS [1, 2]

- ExBLAS is an accurate & reproducible BLAS based on floating-point expansions (FPE) with error-free transformations (EFT: twosum and twoprod) and Kulisch accumulator
- Assures reproducibility through correct-rounding by preserving every bit of information until the final rounding
- CPU and GPU (OpenCL) versions



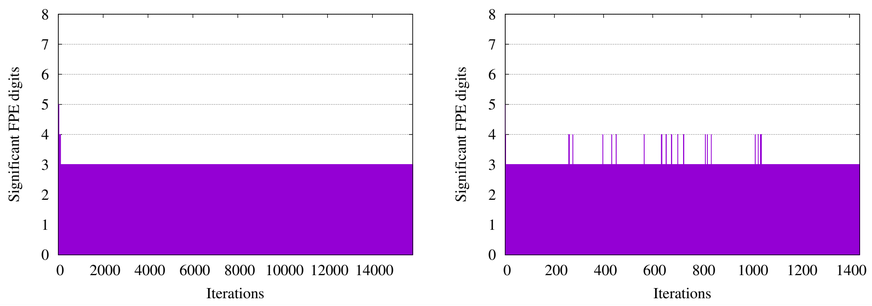
OzBLAS [3, 4]

- OzBLAS is an reproducible & accurate BLAS using the Ozaki scheme, which is an accurate matrix multiplication method based on the error-free transformation of dot-product
- The accuracy is tunable and depends on the range of the inputs and the vector length
- CPU and GPU (CUDA) versions



Error-free transformation of dot product

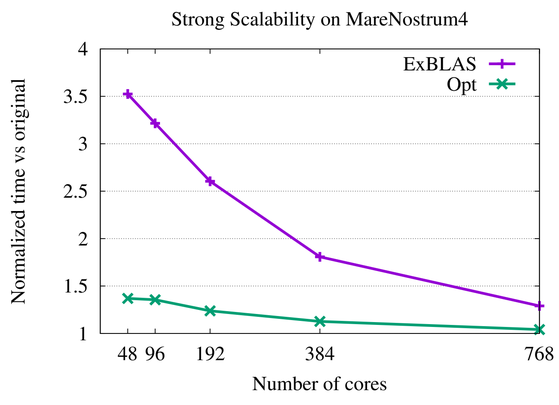
The *third approach (called Opt)* is a customized version of ExBLAS that *relies upon FPE with EFT* [2] as often it is sufficient to have several digits, see gyro_k on left and msc01050 on right.



Performance Results

We provide implementations on different CPUs and GPUs and verify the accuracy of our implementations against the (P)CG with the multiple precision MPFR library. This table shows results on MN4@BSC, where *Opt* stands for FPE with eight digits and the early-exit technique, i.e. omit propagating zeros. The ExBLAS & Opt results are identical for various number of MPI processes as well as on both MN4 and TR clusters.

Iteration	Absolute Residual Norm ($\|\|\boldsymbol{r}_i\|\|$)		
	MPFR	*MPI 48 processes*	***ExBLAS & Opt***
0	0x1.19f179eb7f032p+49	0x1.19f179eb7f03**3**p+49	0x1.19f179eb7f032p+49
2	0x1.f86089ece9f75p+38	0x1.f86089ece**af76**p+38	0x1.f86089ece9f75p+38
9	0x1.fc59a29d329ffp+28	0x1.fc59a29d32**d1b**p+28	0x1.fc59a29d329ffp+28
10	0x1.74f5ccc211471p+22	0x1.74f5ccc2**01246**p+22	0x1.74f5ccc211471p+22
...	...	...	...
40	0x1.7031058eb2e3ep-19	0x1.7031058e**af4c2**p-19	0x1.7031058eb2e3ep-19
42	0x1.4828f76bd68afp-23	0x1.4828f76bd**a71a**p-23	0x1.4828f76bd68afp-23
45	0x1.8646260a70678p-26	0x1.8646260a**6da06**p-26	0x1.8646260a70678p-26
47	0x1.13fa97e2419c7p-33	0x1.13fa97e24**0f7c**p-33	0x1.13fa97e2419c7p-33



Relative residual $||\boldsymbol{r}_i||/||\boldsymbol{b}||$ (solid), relative true residual $||\boldsymbol{b} - \boldsymbol{Ax}_i||/||\boldsymbol{b}||$ (dotted); GPU1 stands for V100 and GPU2 for P100.

We tested the ExBLAS (FP64Ex-CR; CR – correctly-rounded) and OzBLAS (FP64Oz-CR) versions of CG on Intel Xeon Gold 6126 (Skylake, 24 cores) and Intel Xeon Phi 7250 (KNL, 68 cores) for the following SuiteSparse matrices with the tolerance 10^{-16}

Matrix	n	nnz	nnz/n	FP64Ex-CR/ FP64		FP64Oz-CR/ FP64			
				Skylake	KNL	Skylake	KNL	V100	P100
tmt_sym	5,080,961	726,713	7.0	54.9	89.5	**28.2**	36.6	17.1	21.7
gridgena	48,962	512,084	10.5	16.9	13.8	11.6	**10.8**	12.6	12.3
cfd1	1,825,580	70,656	25.8	22.3	17.0	15.1	**11.7**	12.3	14.9
cbuckle	13,681	676,515	49.4	11.1	**6.4**	16.0	11.5	9.7	10.3
BenElechi1	245,874	13,150,496	53.5	**15.8**	41.2	16.7	18.3	12.8	16.9
gyro_k	17,361	1,021,159	58.8	9.5	**7.5**	12.7	10.1	7.6	8.5
pdb1HYS	36,417	4,344,765	119.3	9.3	12.6	13.1	**8.1**	6.0	6.9
nd24k	72,000	28,715,634	398.8	**5.0**	22.1	13.8	12.1	5.5	6.4

Conclusions and Future Work

- Combined algorithmic and programming strategies led to accurate and reproducible Conjugate Gradient on CPUs, GPUs, and cross-platforms
- ExBLAS- and FPE-based PCG show only 20 % and 5 % overhead, respectively, on up to 764 cores but requires development of BLAS routines
- Ozaki-scheme-based CG is tunable and efficient and it relies on vendor-provided BLAS

We explore mixed-precision scenarios as well as a more optimized FPE-based PCG variant.

Acknowledgements

This research was partially supported by JSPS KAKENHI Grant No.19K20286 and the EU H2020 MSCA-IF Robust Grant No. 842528.

References

[1] S. Collange, D. Defour, S. Graillat, and R. Iakymchuk. Numerical reproducibility for the parallel reduction on multi- and many-core architectures. *ParCo*, 49:83–97, 2015.

[2] R. Iakymchuk, M. Barreda, S. Graillat, J. I. Aliaga, and E. S. Quintana-Ortí. Reproducibility of Parallel Preconditioned Conjugate Gradient in Hybrid Programming Environments. *IJHPCA*, 34(5):502–518, 2020.

[3] D. Mukunoki, T. Ogita, and K. Ozaki. Accurate and reproducible blas routines with ozaki scheme for many-core architectures. In *Proc. of PPAM2019. Lecture Notes in Computer Science*, volume 12043, pages 516–527, 2020.

[4] D. Mukunoki, K. Ozaki, T. Ogita, and R. Iakymchuk. Conjugate Gradient Solvers with High Accuracy and Bit-wise Reproducibility between CPU and GPU using Ozaki scheme. In *Proc. of HPC Asia*, pages 100–109, 2021.

code: `https://github.com/riakymch/ReproCG`
`http://www.math.twcu.ac.jp/ogita/post-k/results.html`
e-mail: `roman.iakymchuk@sorbonne-universite.fr`