

An Error Correction Solver for Linear Systems: Evaluation of Mixed Precision Implementations

Hartwig Anzt
Vincent Heuveline
Björn Rocker

No. 2010-01

Preprint Series of the Engineering Mathematics and Computing Lab (EMCL)





Preprint Series of the Engineering Mathematics and Computing Lab (EMCL)
ISSN 2191-0693
No. 2010-01

Impressum

Karlsruhe Institute of Technology (KIT)
Engineering Mathematics and Computing Lab (EMCL)

Fritz-Erler-Str. 23, building 01.86
76133 Karlsruhe
Germany

KIT – University of the State of Baden Wuerttemberg and
National Laboratory of the Helmholtz Association

Published on the Internet under the following Creative Commons License:
<http://creativecommons.org/licenses/by-nc-nd/3.0/de> .



www.emcl.kit.edu

An Error Correction Solver for Linear Systems: Evaluation of Mixed Precision Implementations

Hartwig Anzt, Björn Rocker and Vincent Heuveline

Karlsruhe Institute of Technology (KIT),
AG Numerical Simulation, Optimization and High Performance Computing,
71628 Karlsruhe

hartwig.anzt@kit.edu, bjoern.rocker@kit.edu,
vincent.heuveline@kit.edu

Abstract. This paper proposes an error correction method for solving linear systems of equations and the evaluation of an implementation using mixed precision techniques .

While different technologies are available, graphic processing units (GPUs) have been established as particularly powerful coprocessors in recent years. For this reason, our error correction approach is focused on a CUDA implementation executing the error correction solver on the GPU. Benchmarks are performed both for artificially created matrices with preset characteristics as well as matrices obtained from finite element discretizations of fluid flow problems.

1 Introduction

The development of modern technology is characterized by simulations, that often are no longer performed through physical experiments, but through mathematical modeling and numerical simulation. In many cases, for example in computational fluid dynamics, massive computation power is needed, in order to handle large systems of linear equations.

For the solving process, often iterative solvers are chosen which can exploit the sparse structure of the affiliated matrix to compute an approximation of a certain accuracy considerably faster than a direct solver.

The computational complexity of this problem depends on the characteristics of the linear system, the properties of the used linear solver and the floating point format.

The floating point format determine not only the execution time when performing computations, but also the occurring rounding errors. Generally, a more complex floating point format usually leads to higher accuracy and higher computational effort. Today, most hardware architectures are configured for the IEEE 754 standard containing single precision and double precision as the main floating point formats. As their names indicate, the double precision format has twice the size of the single precision format, leading to a factor of two in computational cost while offering a higher precision.

In many cases single precision floating point operations are not suitable for scientific computation. The question arises, whether the whole algorithm has to be performed in the double precision format, or whether one can gain speed by computing parts of it in single precision and other parts in double precision, and still obtain double precision accuracy for the final result.

One approach is to modify the algorithm of an error correction method such that the inner error correction solver uses a lower format than the working precision. As the final accuracy only depends on the stopping criterion of the refinement solver, the solution approximation is not affected. Still, it can be expected that the mixed precision approach performs faster than a plain solver in high precision, since the cheaper error correction solver in the low precision format may overcompensate the additional computations and typecasts.

2 Mixed Precision Error Correction Methods

2.1 Mathematical Background

Error correction methods have been known for more than 100 years, and have finally become of interest with the rise of computer systems in the middle of the last century. The core idea is to use the residual of a computed solution as the right-hand side to solve a correction equation.

The motivation for the error correction method can be obtained from Newton's method. Newton developed a method for finding successively better approximations to the zeros of a function $f(\cdot)$ by updating the solution approximation x_i through

$$x_{i+1} = x_i - (\nabla f(x_i))^{-1} f(x_i). \quad (1)$$

We now apply Newton's method (1) to the function $f(x) = b - Ax$ with $f'(x) = A$. By defining the residual $r_i := b - Ax_i$, we obtain

$$\begin{aligned} x_{i+1} &= x_i - (\nabla f(x_i))^{-1} f(x_i) \\ &= x_i + A^{-1}(b - Ax_i) \\ &= x_i + A^{-1}r_i. \end{aligned}$$

Denoting the solution update with $c_i := A^{-1}r_i$, we can design an algorithm.

```

1: initial guess as starting vector:  $x_0$ 
2: compute initial residual:  $r_0 = b - Ax_0$ 
3: while ( $\| Ax_i - b \|_2 > \varepsilon \| r_0 \|$ ) do
4:    $r_i = b - Ax_i$ 
5:   solve:  $Ac_i = r_i$ 
6:   update solution:  $x_{i+1} = x_i + c_i$ 
7: end while

```

Algorithm 1: Error Correction Method

Here x_0 is an initial guess. In each iteration, the inner correction solver searches for a c_i , such that $Ac_i = r_i$ with r_i being the residual of the solution approximation x_i . Then, the approximation of the solution x_i is updated to $x_{i+1} = x_i + c_i$.

2.2 Mixed Precision Approach

The underlying idea of mixed precision error correction methods is to use different precision formats within the algorithm of the error correction method, updating the solution approximation in high precision, but computing the error correction term in lower precision. This approach was also suggested by [1],[2], [3], and [4].

Hence, one regards the inner correction solver as a black box, computing a solution update in lower precision. The term high precision refers to the precision format that is necessary to display the accuracy of the final solution and we can obtain the following algorithm where \cdot^{high} denotes the high precision value and \cdot^{low} denotes the value in low precision. The conversion between the formats will be left abstract throughout this paper. Because especially the conversion of the matrix A is expensive, it should be stored in both precision formats, high and low precision. In the case of using hybrid hardware, A should be stored in the local memory of the hardware devices in the respectively used format.

Using the mixed precision approach to the error correction method, we have to be aware of the fact that the residual error bound of the error correction solver may not exceed the accuracy of the lower precision format. Furthermore, each error correction produced by the inner solver in lower precision cannot exceed the data range of the lower precision format. This means that the smallest possible error correction is the smallest number ϵ_{low} , that can be represented in the lower precision. Thus, the accuracy of the final solution cannot exceed ϵ_{low} either. This can become a problem when working with very small numbers, because then the solution correction terms can not be denoted in low precision, but in most cases, the problem can be avoided by converting the original values to a lower order of magnitude.

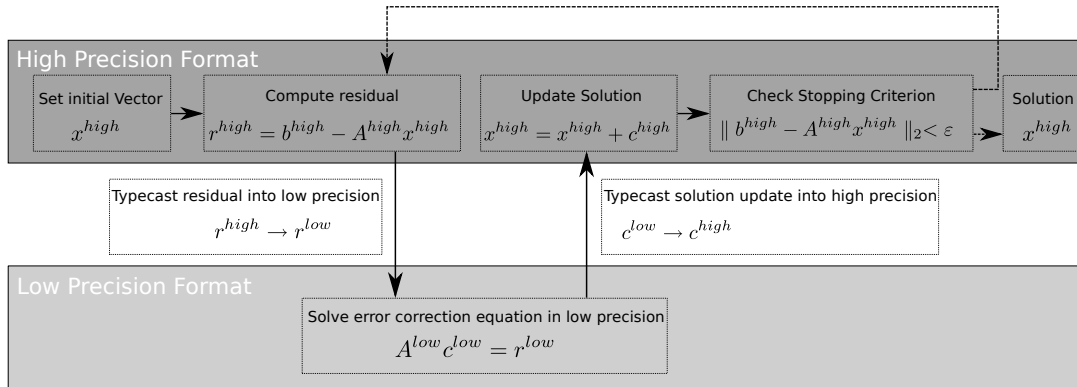


Fig. 1: Visualizing the mixed precision approach to an error correction Solver

Using the displayed algorithm we obtain a mixed precision solver. If the final accuracy does not exceed the smallest number that can be represented in the lower precision, it gives exactly the same solution approximation as if the solver was performed in the high precision format. Theoretically, any precision can be chosen, but in most cases it is comfortable to use the IEEE 754 standard formats.

The computation of the correction loop $A^{low}c^{low} = r^{low}$ can be performed with a direct solver, or again with an iterative method. This implies that it is even possible to cascade a number of error correction solvers using decreasing precision.

In the case of an iterative solver as error correction solver, especially the iterative approaches to the Krylov subspace methods are of interest, since these provide an approximation of the residual error iteratively in every computation loop. Hence, one is able to set a certain relative residual stopping criterion for the iterative error correction solver. Possible Krylov subspace solvers include the CG Algorithm, GMRES, CGSTAB etc. The mixed precision error correction method based on a certain error correction solver poses the same demands to the linear problem, as the within used Krylov subspace solver.

In the case of a direct error correction solver, the solution update usually has a quality depending on the condition number of the system and the lower precision format. Hence, the solution improvement normally depends on the system, but is generally high. Despite the fact that direct methods are computationally expensive, they are of interest as error correction solver, since some of them own pleasant characteristics:

Using for example the LU solver as error correction solver, the LU decomposition has to be computed only in the first error correction loop. In the following loops, the stored decomposition can be used to perform the forward and backward substitution. Since these substitutions imply only low computational effort, they can, depending on the hardware structure, even be performed in the high precision format. This leads to accuracy advantages and economizes the algorithm by omitting the computationally expensive typecasts of the residual and the solution update.

It should be mentioned, that the solution update of the error correction solver is usually not optimal for the outer system, since the discretization of the problem in the lower precision format contains rounding errors, and it therefore solves a perturbed problem. When comparing the algorithm of an error correction solver to a plain solver, we realize, that the error correction method has more computations to execute. Each outer loop consists of the computation of the residual error term, a typecast, an initialization of a vector, the scaling process, the inner solver for the correction term, the reconversion of the data and the solution update. The computation of the residual error itself consists of a matrix-vector multiplication, a vector addition and a scalar product. Using a hybrid architecture, the converted data additionally has to be transmitted between the devices.

The mixed precision refinement approach to a certain solver is superior to the

plain solver in high precision, if the additional computations and typecasts are overcompensated by the cheaper inner correction solver using a lower precision format.

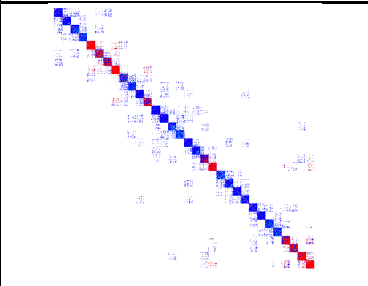
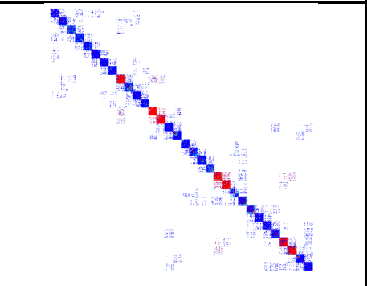
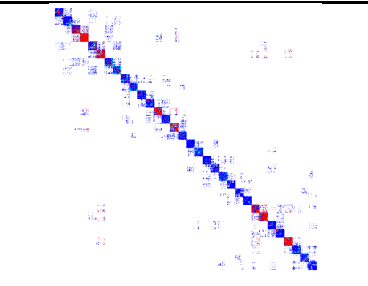
3 Numerical Experiments

To be able to compare the performance of different implementation of the GMRES-(10) solver, we perform tests with different linear systems. In this work 10 denotes the restart parameter for the GMRES.

All solvers use the relative residual stopping criterion $\varepsilon = 10^{-10} \| r_0 \|_2$. Due to the iterative residual computation in the case of the plain GMRES-(10) solvers, the mixed GMRES-(10) solvers based on the mixed precision error correction method usually iterate to a better approximation since they compute the residual error explicitly, but as the difference is generally small, the solvers are comparable. In case of the mixed precision GMRES-(10) on the TESLA-System, the error correction solver is performed on the GPU, while the solution update is led to the CPU of the same system. This is done to be able to handle larger problems since the amount of memory on the GPU is limited to 4 GB.

On the one hand, we use matrices with a preset condition number, preset sparsities, and increase the dimension. Depending on the sparsity, the matrices are stored in the matrix array storage format (MAS) or the compressed row storage format (CRS).

On the other hand, we additionally use linear problems that were obtained out of a discretization of CFD. The three systems of linear equations CFD1, CFD2 and CFD3 are affiliated with the 2D modeling of a Venturi Nozzle in different discretization fineness. The distinct number of supporting points yields to different matrix characteristics concerning the dimension, the number of non-zeros, and the condition number.

CFD1	CFD2	CFD3
		
problem: 2D fluid flow problem size: $n = 395009$ sparsity: $nnz = 3544321$ storage format: CRS	problem: 2D fluid flow problem size: $n = 634453$ sparsity: $nnz = 5700633$ storage format: CRS	problem: 2D fluid flow problem size: $n = 1019967$ sparsity: $nnz = 9182401$ storage format: CRS

Tab. 1: Sparsity plots and properties of the CFD test-matrices

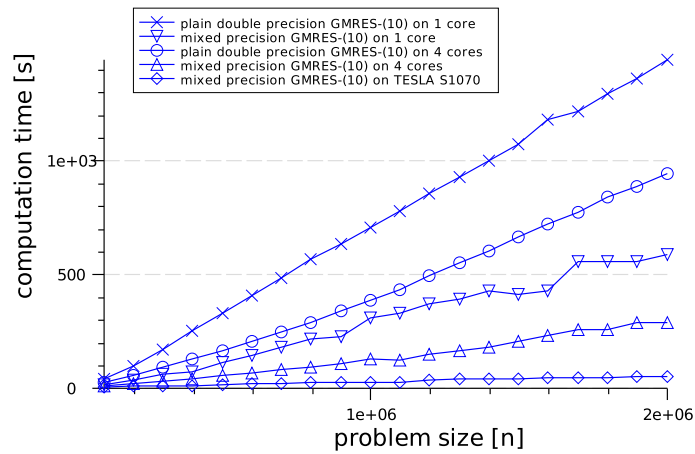


Fig. 2: Test case: sparse linear system, condition number $\kappa \approx 8000$, relative residual stopping criterion $\varepsilon = 10^{-10}$;

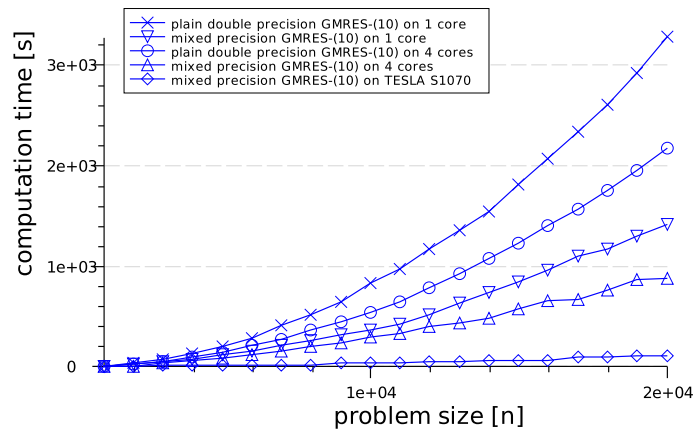


Fig. 3: Test case: sparse linear system, condition number $\kappa \approx 8000$, relative residual stopping criterion $\varepsilon = 10^{-10}$;

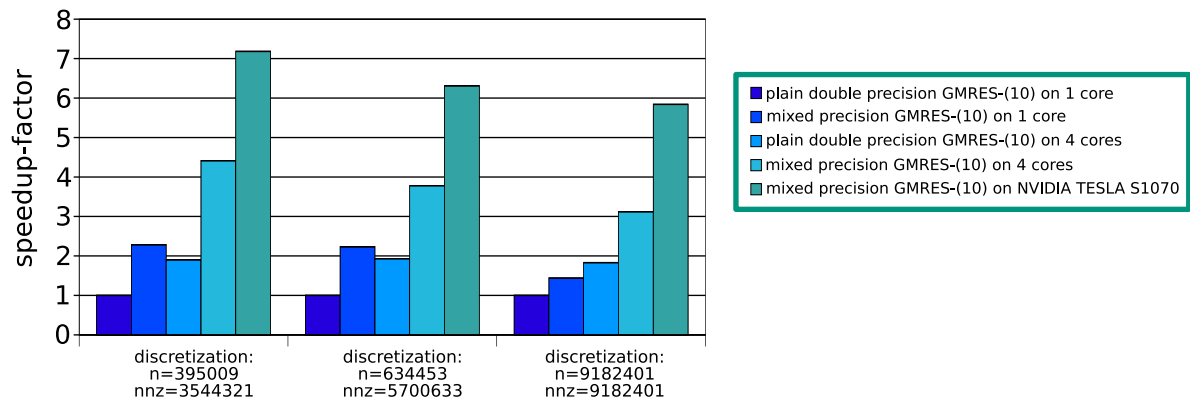


Fig. 4: Speedup of the different solvers for the CFD simulation of the Venturi Nozzle in different discretization fineness; $\varepsilon = 10^{-10}$; $\varepsilon_{\text{inner}} = 0.1$;

CFD1

solver type	computation time [s]
sequential double GMRES-(10) on IC1	3146.61 sec
sequential mixed precision GMRES-(10) on IC1	1378.56 sec
parallel double GMRES-(10) on IC1 using 4 cores	1656.53 sec
parallel mixed precision GMRES-(10) on IC1 using 4 cores	712.83 sec
mixed precision GMRES-(10) on NVIDIA TESLA S1070	438.13 sec

CFD2

solver type	computation time [s]
sequential double GMRES-(10) on IC1	13204.70
sequential mixed precision GMRES-(10) on IC1	5924.32
parallel double GMRES-(10) on IC1 using 4 cores	6843.66
parallel mixed precision GMRES-(10) on IC1 using 4 cores	3495.09
mixed precision GMRES-(10) on NVIDIA TESLA S1070	2092.84

CFD3

solver type	computation time [s]
sequential double GMRES-(10) on IC1	60214.50
sequential mixed precision GMRES-(10) on IC1	41927.40
parallel double GMRES-(10) on IC1 using 4 cores	32875.10
parallel mixed precision GMRES-(10) on IC1 using 4 cores	19317.00
mixed precision GMRES-(10) on NVIDIA TESLA S1070	10316.70

4 Hardware Platform and Implementation Issues

The utilized TESLA-System is equipped with one NVIDIA TESLA S1070. The two host nodes, each connected via a PCIe 2.0 x16 to the S1070, are each equipped with two Intel Xeon 5450 CPUs. The Intel MKL in version 10.1.1.019 and the Intel compiler in version 11.0.074 was used.

The InstitutsCluster is located at the Karlsruhe Institute of Technology (KIT) and consists of 200 computing nodes each equipped with two Intel quad-core EM64T Xeon 5355 processors, owning 16 GB of main memory. Peak performance of one node is about 85,3 GFlops. For a performance evaluation see [5]. The software platform is the Intel CMKL in version 10.1.2.024 and the Intel compiler in version 10.1.022.

5 Conclusions and Future Work

The numerical tests in this paper have shown the high potential of using different precision formats within the proposed error correction solver.

The obtained algorithm is flexible in terms of choosing the inner correction solver, and robust in terms of numerical stability. The possibility of performing the error correction solver on a coprocessor increases the potential of mixed

precision methods, as they can be implemented efficiently on hybrid systems. Performing the error correction solver of an error correction method in a lower format leads to an overall increase in performance for a large number of problems.

On a CPU, performing the error correction method in mixed precision, one often achieves a speedup factor of two compared to the plain solver in double precision. When using hybrid hardware, consisting of coprocessors specialized on low precision performance, even higher speedup factors can be expected. In the numerical tests for the FEM discretization of the Venturi Nozzle we achieved speedups of more than seven. Still, a very ill-conditioned problem can lead to a high number of additional outer iterations necessary to correct the rounding errors, that arise from the use of a lower precision format in the error correction solver. Due to the fact that we are usually not able to determine a priori whether the mixed precision method is superior for a specific problem, an optimized implementation of the solver would execute the first solution update of the mixed precision error correction method and determine, depending on the improvement of the solution approximation, whether it should continue in the mixed precision mode or whether it should use the plain solver in high precision. The next step beyond this strategy of changing between single and double precision is to use techniques around adaptive precision, where the precision is adjusted according to the convergence in the inner solver. FPGAs and related technologies will in the future provide the capabilities for such algorithms.

For an efficient implementation of the mixed precision error correction techniques in a solver suite, some additional work is necessary, especially concerning the use of preconditioners. Adding a preconditioner to a solver increases not only the stability of the solver, but also its performance. In such an environment, the mixed precision error correction methods form powerful solvers for FEM simulations.

References

1. D. Göddeke, R. Strzodka, and S. Turek. Performance and accuracy of hardware-oriented native-, emulated- and mixed-precision solvers in FEM simulations. 2007.
2. D. Göddeke and R. Strzodka. Performance and accuracy of hardware-oriented native-, emulated- and mixed-precision solvers in FEM simulations (part 2: Double precision gpus). Technical report, Fakultät für Mathematik, TU Dortmund, 2008.
3. Marc Baboulin, Alfredo Buttari, Jack J. Dongarra, Julie Langou, Julien Langou, Piotr Luszczek, Jakub Kurzak, and Stanimire Tomov. Accelerating scientific computations with mixed precision algorithms. 2008.
4. Alfredo Buttari, Jack J. Dongarra, Julie Langou, Julien Langou, Piotr Luszczek, and Jakub Kurzak. Mixed precision iterative refinement techniques for the solution of dense linear systems. 2007.
5. V. Heuveline, B. Rocker, and S. Ronnas. Numerical simulation on the sicortex supercomputer platform: a preliminary evaluation. In *EMCL Preprint Series*, accepted.

Preprint Series of the Engineering Mathematics and Computing Lab

recent issues

- No. 2009-02 Rainer Buchty, Vincent Heuveline, Wolfgang Karl, Jan-Philipp Weiß: A Survey on Hardware-aware and Heterogeneous Computing on Multicore Processors and Accelerators
- No. 2009-01 Vincent Heuveline, Björn Rucker, Staffan Ronnas: Numerical Simulation on the SiCortex Supercomputer Platform: a Preliminary Evaluation