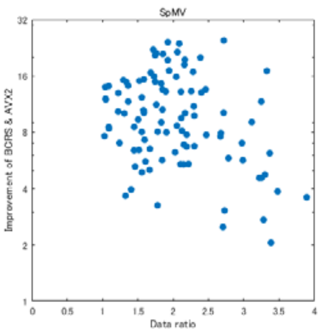
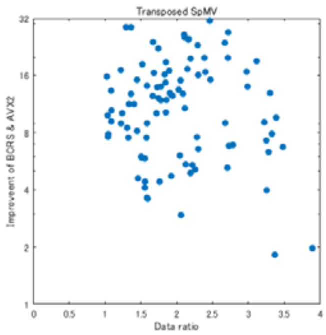
Robust and Fast BiCG using SIMD accelerated DD arithmetic

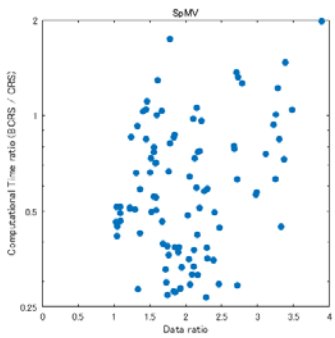
Hidehiko Hasegawa, Toshiaki Hishinuma
University of Tsukuba, Japan

Fast DD-precision SpMV and transposed SpMV on AVX2

・**DD arithmetic**[1] uses two double precision variables to implement one quadruple-precision variable. A DD addition consists of 11 double-precision additions, and a DD multiplication consists of 15 double-precision additions and 9 double-precision multiplications.
・**Intel AVX2** is SIMD function, and can process 4 double-precision operations simultaneously.
・We add dummy operations to a sparse matrix in **C**ompressed **R**ow **S**torage format and extra data to a sparse matrix in **B**locked **C**ompressed **R**ow **S**torage format[2] to perform 4 double-precision operations simultaneously. Each block consists of 4 rows and 1 column.
・We use **double-precision for Matrices** to reduce memory space and the bytes/flops of SpMV. The bytes/flops are 1.33(28/21) for this, 10(20/2) for the ordinary double-precision operations, and 1.56(28/21) for full DD-precision[3].
・BCRS(4,1) needs to store extra zero elements for the matrix, however it has good performance improvement on AVX2.
(Data ratio means (# of Data in BCRS(4,1)) / (# of Data in CRS); Improvement means (Time of CRS without AVX2)/(Time of BCRS(4,1) with AVX2))







Mixed precision arithmetic Double-precision and DD-precision

The computation cost of DD-precision SpMV and transposed SpMV on AVX2 is approximately 3 times of that of ordinary double-precision computation. This means "good accuracy" but "costly". We should try to reduce total computation cost of iterative solvers as a "Hybrid".

(1) Combination of Double and DD precisions in each iteration step
(2) DQ-SWITCH [4]
- ✓Current solution x_k is passed at the restart
- ✓ Upper and Lower part of DD-precision variables are stored in different arrays
- ✓ Only Upper part is used for Double Precision
- ✓ Two Steps are performed by Different Precision

(3) Automatic restart for DQ-SWITCH
- ✓ Compute deviation of residual norm and restart at

$$v = \frac{1}{p}\sum_{i=1}^{p}\left(\frac{nrm(i) - nrm(1)}{nrm(1)}\right)^2$$

Diverge : $v \geq 10^2$
Stagnate : $v \leq 10^{-1}$

```
for(k=0;k<matitr;k++){
   The first step
   if( nrm2<restart_tol ) break;
}
Clear all values except x
for(k=k+1;k<maxtr;k++) {
   The second step
   if( nrm2<tol ) break;
}
```

(4) Full DD-precision
(Testing bed: 4 nodes, intel Core i7 4770 4core 3.4GHz (8MB, 16GB), Fedora21, intel C/C++ Compiler 13.0.1)

Computation Result

	ASIC_100ks (N = 99,190)	TSOPF_RS_b39_c7 (N = 141,098)	memplus (N = 17,758)	epb3 (N = 84,617)
All Double	3371(3.2s)	6204(2.5s)	∞	∞
p : DD	3156(3.8s)	4043(1.7s)	∞	∞
p*: DD	3693(4.5s)	5789(2.4s)	12129(5.0s)	∞
p and p*: DD	3240(4.2s)	3871(1.9s)	11613(5.7s)	13528(50.8s)
Vectors : DD	3011(2.7s)	3646(1.8s)	10938(5.4s)	10432(35.9s)
Full DD	3011(5.8s)	3646(4.1s)	10938(12.3s)	10434(78.8s)
DQ-SWITCH	3036(2.8s)	3863(2.0s)	11589(6.1s)	11756(33.2s)

Conclusions

・Partial use of DD-precision in each iteration has small improvement, however sometimes does not converge.
・DD-precision except matrices is robust same as to Full DD-precision and reasonable computation time.
・**DQ-SWITCH** may have small improvement in keeping robustness.
・Automatic restart is not easy. Especially, **BiCG** has no special property to detect its restart.
・**Mixed precision iterative methods** are practically useful, because they are **robust** and **fast**. They also have parallelism in original algorithms.
・Incorporating preconditioning is one of our future works. For introducing preconditioning, there are some choices such as which part, precisions, and kind of preconditioning.

References

[1] Bailey, D, H.: High-Precision Floating-Point Arithmetic in Scientific Computation, computing in Science and Engineering, pp. 54-61 (2005); [2] Barrett, R., et al.: Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, SIAM, pp. 57-65 (1994); [3] Hishinuma, T., et al. : SIMD Parallel Sparse Matrix-Vector and Transposed-Matrix-Vector Multiplication in DD Precision, VECPAR2016, LNCS 10150, pp.1-14 (2017).; [4] Kotakemori, H., et al.: Implementation of Fast Quad Precision Operation and Acceleration with SSE2 for Iterative Solver Library, IPSJ Transactions on Advanced Computing Systems, 1(1), pp. 73-84 (2008). (in Japanese); [5] The University of Florida Matrix Collection;

Contact : hasegawa@slis.tsukuba.ac.jp