Operator Splittings on GPUs

Christoph Klein **Robert Strzodka**

Institute of Computer Engineering (ZITI), Heidelberg University

Operator splittings are a successful method for the solution of parabolic partial differential equations. We apply the same ideas for general sparse linear equation systems. For general graphs, which are induced by the sparse matrix of the equation system, a parallel segmentation algorithm is used to extract one-dimensional segments, and create appropriate renumberings, such that the permuted subgraph has a tridiagonal form. With these tridiagonal factors, multiplicative and alternating operator splittings are created and applied as preconditioners in iterative solvers while leveraging our high-performance tridiagonal GPU solver.

Tridiagonal GPU Solver with Scaled Partial Pivoting

We propose the **R**ecursive **P**artitioned **T**ridiagonal **S**chur Complement Algorithm (RPTS)[2], which is implemented as a hierarchical tridiagonal GPU solver with scaled partial pivoting. It reads $8N$ and writes N matrix elements in the first stage, and is only limited by memory bandwidth. The time which is spent in subsequent stages of the hierarchically built solver is negligible small because the read data size is less than 4% of the original size.

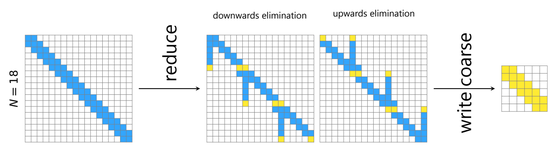

Figure 1. Reduction step of RPTS to obtain a coarser tridiagonal system.

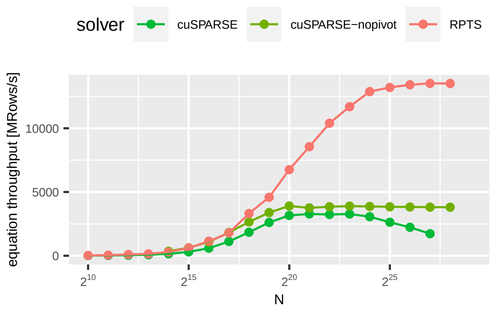

Figure 2. Single precision tridiagonal solver performance on a GeForce RTX 2080 Ti.

Alternating and Multiplicative Operator Splittings

In solutions of a sparse linear system $Ax = d$ a preconditioner iteration is an important building block.

$$x^{k+1} = x^k + M^{-1}(d - Ax^k), \tag{1}$$

where M denotes the preconditioner. M is often chosen based on a tradeoff between computation time and improvement of the residual. We choose a multiplicative operator splitting ansatz

$$M_{\text{MOS}} := T \prod_{l=m-1}^{0} M_l, \tag{2}$$

with an invertible diagonal matrix T and invertible matrices M_l. Moreover, we make the alternating ansatz, where the iterations may alternate between different splittings $\bar{M}_0, \ldots, \bar{M}_{m-1}$ resulting in the repetition $(k = 0, \ldots, k_{\text{last}} - 1)$ of the alternating corrections

$$\begin{aligned} x^{mk+1} &:= x^{mk+0} + \bar{M}_0^{-1}(b - Ax^{mk+0}) \\ &\vdots \\ x^{mk+m} &:= x^{mk+m-1} + \bar{M}_{m-1}^{-1}(b - Ax^{mk+m-1}), \end{aligned} \tag{3}$$

with alternating splittings $\bar{M}_0, \ldots, \bar{M}_{m-1}$.

With the appropriate preconditioner $\bar{M}_{\text{ALT-i}}$ the alternating corrections can be expressed as a single correction

$$\bar{M}_{\text{ALT-i}} := A(I - \bar{G}_{\text{ALT-i}})^{-1}, \quad \bar{G}_{\text{ALT-i}} := \prod_{l=m-1}^{0} (I - \bar{M}_l^{-1}A) \tag{4}$$

$$x^{k+1} := x^k + \bar{M}_{\text{ALT-i}}^{-1}(b - Ax^k). \tag{5}$$

For the multiplicative ansatz (Eq. 2), we choose two approaches. First, a **direct** approach, which is constructed by

$$E := \frac{1}{m} T^{-1}(\text{diag}(A) - T) \tag{6}$$

$$J := I + E \tag{7}$$

$$M'_l := (TE + A'_l), \quad A'_l := J^{-(m-1-l)} A''_l J^{-l} \tag{8}$$

$$M_l := (I + T^{-1}M'_l) = (J + T^{-1}A'_l), \tag{9}$$

The A''_l are off-diagonal matrices, which are extracted from A. With

$$T_{i,i} := (\text{diag}_{dl1}(A))_{i,i} := \text{unit}(A_{i,i}) \max\left(|A_{i,i}|, \sum_{j:j\neq i} |A_{i,j}|\right) \tag{10}$$

$$\text{unit}(x) := \begin{cases} 1 & \text{if } x = 0 \\ x/|x| & \text{else}, \end{cases} \tag{11}$$

we *choose* T based on an error analysis derived from matrix perturbation theory. With the latter, we obtain the **direct** multiplicative operator splitted preconditioner

$$M_{\text{MOS-d}} := T \prod_{l=m-1}^{0} (I + T^{-1}M'_l) = A + R, \tag{12}$$

and R representing the perturbation.
For the **adaptive** construction, we begin with the MOS ansatz

$$M_{\text{MOS-a}} := T \prod_{l=m-1}^{0} M_l, \tag{13}$$

but now the M_l are constructed recursively with intermediate pruning:

$$\begin{aligned} B_0 &:= \texttt{prune}(A, S_0^B) & M_0 &:= \texttt{prune}(B_0, S_0^M) \\ B_1 &:= \texttt{prune}(B_0 M_0^{-1}, S_1^B) & M_1 &:= \texttt{prune}(B_1, S_1^M) \\ &\vdots \\ B_{m-1} &:= \texttt{prune}(B_{m-2} M_{m-2}^{-1}, S_{m-1}^B) & M_{m-1} &:= \texttt{prune}(B_{m-1}, S_{m-1}^M) \\ B_m &:= \texttt{prune}(B_{m-1} M_{m-1}^{-1}, S_m^B) & T &:= \texttt{diag}_{dl1}(B_m) \\ & & S_l^B &\supseteq S_l^M. \end{aligned} \tag{14}$$

the S_l^B, S_l^M are sparsity patterns, where the S_l^B are mainly limited by computational and memory limitations.

Special Case of Tridiagonal Factors

So far the construction of the $\bar{M}_l$ (Eq. 3), the A''_l (Eq. 9), and the S_l^M (Eq. 14) was not specified. Here, we construct tridiagonal factors based on one-dimensional segmentations of the graph, which is induced by the matrix A or B_l.

To find one-dimensional segments in a graph, **two** possible segment connections are proposed along existing edges in the graph according to some heuristic, e.g. strongest weights (Figure 3). This can be done in parallel for each vertex.

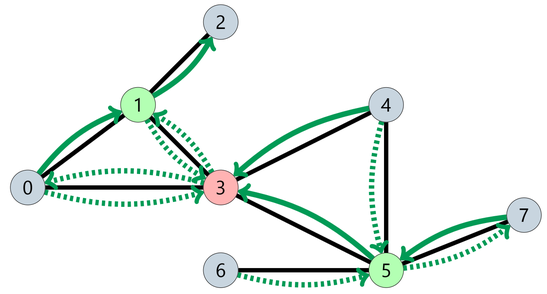

Figure 3. Colored graph with proposed (dashed green) and confirmed (solid green) segment connections.

Once all vertices belong to some segment the vertices can be renumbered such that neighboring vertices in the segment are consecutive in the numbering.

Note, that every M_l is of tridiagonal form. Between the inversion of these tridiagonal system with the RPTS algorithm the right-hand side vector must be permuted once to have the same ordering as the corresponding M_l.

Exemplaric Test Cases with Tridiagonal Factors

We apply a single precision GMRES iterative solver within a double precision iterative refinement and generate the solution with $\hat{x}_i := \sin(16\pi i/N)$. The outer iterative defect correction in double precision is executed after 20 iterations, which is also the restart parameter of the GMRES. The results were done on a machine with CentOS 7, CUDA Toolkit 11.2.142, CUDA driver 450.57, host compiler GCC 10.2.0, and a GeForce RTX 2080 Ti. Both test cases are finite element problems, where the first one is a steady state **thermal** problem, and the second one is a density dependent **transport** problem in groundwater. The corresponding sparse matrices are also available in the Sparse Matrix Collection. We compare against the ILU-ISAI implementation in Magma by Anzt et al. [1] and use also the GMRES implementation from Magma.

Matrix	DOFs	nnz
TRANSPORT	1 602 111	23 500 731
THERMAL2	1 228 045	8 580 313
THERMAL1	82 654	574 458

Results

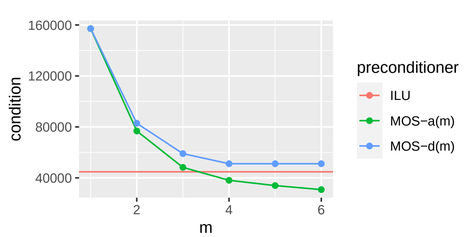

Figure 4. Condition numbers of $M^{-1}A$ determined with power iteration for matrix THERMAL1 and different preconditioners.

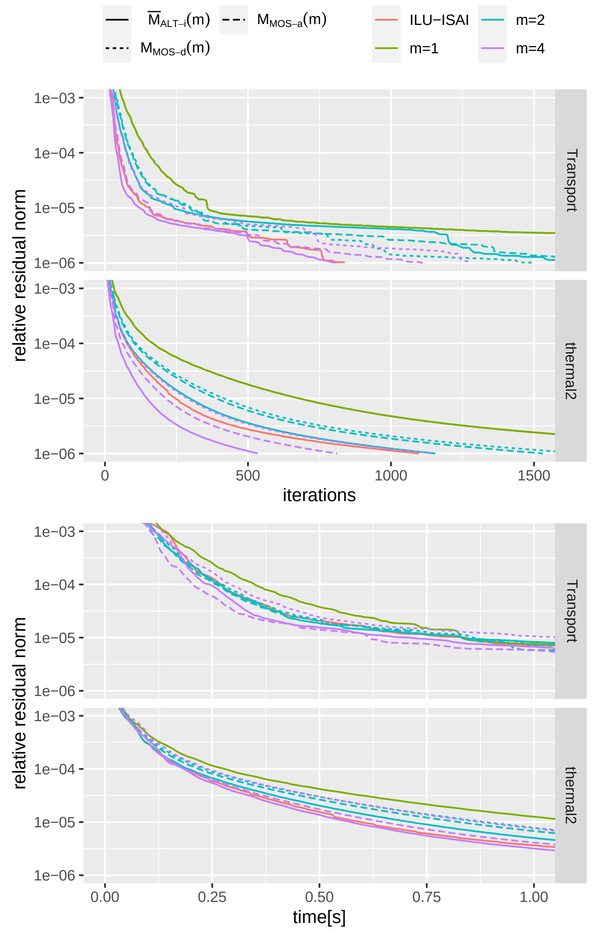

Figure 5. Convergence for different preconditioners with GMRES in mixed precision iterative refinement. All curves start at (0,1).

For the shown test cases, the operator splitted and alternating preconditioners have the desirable property that they improve in condition and convergence monotonically with m (Figure 4 and 5).

Figure 5 shows that already our tridiagonal preconditioners keep up with state of the art high-performance preconditioners like the ILU-ISAI.

References

[1] Hartwig Anzt, Thomas K. Huckle, Jürgen Bräckle, and Jack Dongarra. Incomplete Sparse Approximate Inverses for Parallel Preconditioning. *Parallel Computing*, 71:1–22, 2018.

[2] Christoph Klein and Robert Strzodka. Tridiagonal GPU Solver with Scaled Partial Pivoting at Maximum Bandwidth. In *50th International Conference on Parallel Processing-ICPP*, pages 1–10, 2021.