Mini Schur Complement preconditioner for Bundle Adjustment

Shrutimoy Das

August 2019

1 Introduction

This article describes the design and implementation of a preconditioner based on an approximation of the global Schur complement. The Hessian that is computed in each Levenberg-Marquardt (LM) iteration can be divided into 4 blocks, 2 of which are the reduced structure matrix block and the reduced camera matrix block. Ideally, these two blocks have a block diagonal structure. Hence, the block Jacobi preconditioner described in literature consists of these two blocks. The preconditioner described in this article approximates the global Schur complement of the Hessian and appends this approximated block to the reduced structure block instead appending the reduced camera block.

2 Structure of the Hessian

The bundle adjustment (BA) problem can be formulated as a non-linear least squares problem. To solve this problem, the Levenberg-Marquardt (LM) algorithm is applied. Each iteration of the LM algorithm requires a system of equations to be solved. This system can be written as

$$(J^T J + \lambda \texttt{diag}(J^T J))\delta = -J^T r \tag{1}$$

where J is a Jacobian, $\lambda\texttt{diag}(J^T J)$ is a regularization term and r is the residual vector. The term $J^T J$ is an approximation of a Hessian and this matrix has a structure as follows

$$H = \left[\begin{array}{c|c} D & L^T \\ \hline L & G \end{array}\right] \tag{2}$$

Here D is block diagonal with each block of size $s \times s$ and G is block diagonal with blocks of size $c \times c$ (Usually, $s = 3$ and c varies between $6 - 9$).
Let the number of points be p and the number of cameras be q. Then $D \in \mathbb{R}^{ps \times ps}$ and $G \in \mathbb{R}^{qs \times qs}$. The L block is a general sparse matrix.

3 Schur Complement Approximation

The global Schur complement S of the Hessian in (2) is given by

$$S = G - LD^{-1}L^T$$

In []several methods for approximation of the global Schur complement have been mentioned. Here , we use the **Mini Schur Complements based on Numbering (MSCN)** scheme is described below.

3.1 Mini Schur Complement based on Numbering (MSCN)

We consider the block 2×2 partitioned system in (2). The matrix H is further partitioned as follows.

$$H = \left[\begin{array}{cc|cc} D_{11} & D_{12} & L_{11}^T & L_{12}^T \\ D_{21} & D_{22} & L_{21}^T & L_{22}^T \\ \hline L_{11} & L_{12} & G_{11} & G_{12} \\ L_{21} & L_{22} & G_{21} & G_{22} \end{array}\right] \tag{3}$$

Now, a further approximation of the matrix in (3) is constructed by dropping the blocks D_{ij}, L_{ij}^T, L_{ij} and G_{ij} for which $i \neq j$. Thus the following approximation $\hat{H}_2$ is obtained.

$$\hat{H}_2 = \left[\begin{array}{cc|cc} D_{11} & & L_{11}^T & \\ & D_{22} & & L_{22}^T \\ \hline L_{11} & & G_{11} & \\ & L_{22} & & G_{22} \end{array}\right] \tag{4}$$

Here, the subscript 2 in $\hat{H}_2$ denotes the number of sub-matrices of the matrix G, namely, G_{11} and G_{22}. The matrix in (4) is further partitioned to get the following matrix

$$\hat{H}_2 = \left[\begin{array}{cc|cc|cc|cc} D_{11} & D_{12} & & & L_{11}^T & L_{12}^T & & \\ D_{21} & D_{22} & & & L_{21}^T & L_{22}^T & & \\ \hline & & D_{33} & D_{34} & & & L_{33}^T & L_{34}^T \\ & & D_{43} & D_{44} & & & L_{43}^T & L_{44}^T \\ \hline L_{11} & L_{12} & & & G_{11} & G_{12} & & \\ L_{21} & L_{22} & & & G_{21} & G_{22} & & \\ \hline & & L_{33} & L_{34} & & & G_{33} & G_{34} \\ & & L_{43} & L_{44} & & & G_{43} & G_{44} \end{array}\right] \tag{5}$$

Again , a sparse approximation of (5) is done by dropping the blocks D_{ij}, L_{ij}^T, L_{ij} and G_{ij} for which $i \neq j$ to obtain the following matrix

$$\hat{H}_4 = \left[\begin{array}{cc|cc|cc|cc} D_{11} & & & & L_{11}^T & & & \\ & D_{22} & & & & L_{22}^T & & \\ \hline & & D_{33} & & & & L_{33}^T & \\ & & & D_{44} & & & & L_{44}^T \\ \hline L_{11} & & & & G_{11} & & & \\ & L_{22} & & & & G_{22} & & \\ \hline & & L_{33} & & & & G_{33} & \\ & & & L_{44} & & & & G_{44} \end{array}\right]$$

Here the subscript 4 in $\hat{H}_4$ denotes the number of sub-matrices of matrix G. Eliminating the blocks L_{ii} by using D_{ii} as a pivot, we obtain an approximation to the global Schur complement S by $\hat{S}_4 = \texttt{blkDiag}(S_{ii})$ where

$$S_{ii} = G_{ii} - L_{ii} D_{ii}^{-1} L_{ii}^T$$

The matrix S_{ii} is called a Mini Schur Complement (MSC). Here, for simplicity , we have partitioned the matrix recursively into a block 2×2 matrix. During implementation, by taking advantage of the sparsity structure of the Hessian and the information about the size of the blocks, we could directly identify the blocks G_{ii} such that S_{ii} is computed as

$$S_{ii} = G_{ii} - L_{ii} D_{ii}^{-1} L_{ii}^T, i = 1 : m$$

where m is the number of MSCs desired. Let $\hat{S_m}$ denote the Schur complement approximation computed from m MSCs. Then the MSC preconditioner can be defined as

$$P = \left[\begin{array}{c|c} D & 0 \\ \hline 0 & \hat{S}_4 \end{array}\right] \tag{6}$$

This preconditioner has a block diagonal structure and does not require much storage space.

4 Implementation

The implementation of the MSC preconditioner and the preconditioned `GMRES` routine for solving the system (1) is done in `C++` by integrating it with the `SSBA` code. The `SSBA` implementation does a column reordering of the matrix in (2) using `COLAMD`. Thus, before calling the `MSC_solve` routine, the right hand side vector, Jt_e, is also permuted using the `LDL_perm` routine. After the `MSC_solve` routine is completed, the resultant solution vector again undergoes an inverse permutation via the `LDL_permt` routine.

Inside the `MSC_solve` routine, the first step is to divide the input matrix (in `CSR` format) into separate blocks, as shown in (2). To maintain the `CSR` format, the `cs_di` structure, available in the `cxsparse`/`csparse` library of `SuiteSparse`

package, is used. Two important parameters used during this stage are the sizes of the D(sizeD) and G(sizeG) blocks. If the number of rows in the Hessian is n, then sizeD = n - sizeG. To derive sizeG, three methods have been used as described below.

1. If N is the number of cameras in the problem, then sizeG = $9 \times N$
2. For smaller problems, domain decomposition is applied.
3. For problems where the D block is not purely block diagonal(some off-diagonal entries are present apart from the block diagonal entries), an iteration is run for all the rows to find the index where D can be extracted as completely block diagonal.

However, it is observed that domain decomposition takes a lot of time for larger matrices. The same problem is observed for method 3. Thus for most of the problems, sizeG is taken as $9 \times N$.

After this step is completed, the construction of the MSCs is done. The number of submatrices in D, G, L^T and L blocks are same. If m is the desired number of MSC blocks, then the submatrices are extracted over m iterations. In each iteration, the submatrices are denoted as : PD for block D, PG for block G, PL for block L and PU for block L^T.

$$\hat{H}_m = \left[\begin{array}{ccc|ccc} PD_{11} & & & PU_{11} & & \\ & \ddots & & & \ddots & \\ & & PD_{mm} & & & PU_{mm} \\ \hline PL_{11} & & & PG_{11} & & \\ & \ddots & & & \ddots & \\ & & PL_{mm} & & & PG_{mm} \end{array}\right]$$

Since L^T is the transpose of block L, PU is taken to be the transpose of PL.This is done using the `umfpack_di_transpose` routine available in `UMFPACK` library from `SuiteSparse`. After this blocks are extracted, the Schur complement for k^{th} iteration is defined as follows.

$$S_{kk} = PG_{kk} - PL_{kk} PD_{kk}^{-1} PU_{kk} \tag{7}$$

To compute $PD_{kk}^{-1} PU_{kk}$ in (7), each column of PU_{kk} is densified to get a vector b. Then the linear system $PD_{kk}x = b$ is solved.The resultant solution is then sparsified to generate columns such that the matrix $AA = PD_{kk}^{-1} PU_{kk}$ is a sparse matrix in `CSR` format. Then $PL_{kk} * AA$ is computed using `cs_di_multiply` and the resultant is subtracted from PG_{kk} using `cs_di_add` to get the MSC of the k^{th} iteration.

4.1 Preconditioner Solve

After the m MSCs have been computed, the Schur approximation is computed as $\hat{S}_m = \texttt{blkDiag}(MSC_i), i = 1 : m$. The matrix $\hat{S}_m$ is of size $sizeG \times sizeG$. Since the D block has already been extracted, the preconditioner is constructed as shown in (6). When a vector y generated by the `GMRES` algorithm is passed as a parameter to the preconditioner solve routine, it is same as solving the following system

$$\begin{bmatrix} D & 0 \\ 0 & \hat{S}_m \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 - LD^{-1}y_1 \end{bmatrix}$$

During implementation, it is done in the following steps:

1. Split vector y as $[y_1; y_2]$ where $y_1 \in \mathbb{R}^{sizeD}$ and $y_2 \in \mathbb{R}^{sizeG}$
2. Compute $z_1 = D^{-1}y_1$
3. Compute $z_2 = \hat{S}_m^{-1}(y_2 - Lz_1)$
4. The solution is returned as $x = [z_1; z_2]$.