Chapter 1

Example: $C := AB^T + BA^T + C$ – Team: Devangi Parikh

1.1 Operation

Consider the operation

$$C := AB^T + BA^T + C$$

where C is a $m \times m$ lower triangular matrix and A and B is a $m \times m$ matrix. This is what we call a symmetric rank-2k update with the Lower triangular matrix on the Right. We will refer to this operation as SYR2K_LN where the LN stands for lower triangular no-transpose.

1.2 Precondition and postcondition

In the precondition

$$C = \widehat{C}$$

$\widehat{C}$ denotes the original contents of C. This allows us to express the state upon completion, the postcondition, as

$$C = C := AB^T + BA^T + \widehat{C}.$$

It is implicitly assumed that C is nonunit lower triangular.

1.3 Partitioned Matrix Expressions and loop invariants

There are two PMEs for this operation.

1.3.1 PME 1

To derive the second PME, partition

$$L \rightarrow \left(\begin{array}{c|c} L_{TL} & 0 \\ \hline L_{BL} & L_{BR} \end{array} \right), \quad \text{and} \quad B \rightarrow \left(\begin{array}{c} B_T \\ \hline B_B \end{array} \right).$$

Substituting these into the postcondition yields

$$\left(\begin{array}{c} B_T \\ \hline B_B \end{array}\right) = \left(\begin{array}{c|c} L_{TL} & 0 \\ \hline L_{BL} & L_{BR} \end{array}\right) \left(\begin{array}{c} \widehat{B}_T \\ \hline \widehat{B}_B \end{array}\right)$$

or, equivalently,

$$\left(\begin{array}{c} B_T \\ \hline B_B \end{array}\right) = \left(\begin{array}{c} L_{TL}\widehat{B}_T \\ \hline L_{BL}\widehat{B}_T + L_{BR}\widehat{B}_B \end{array}\right),$$

which we refer to as the first PME for this operations.

From this, we can choose two loop invariants:

Invariant 1: $\left(\begin{array}{c} B_T \\ \hline B_B \end{array}\right) = \left(\begin{array}{c} \widehat{B}_T \\ \hline L_{BR}\widehat{B}_B \end{array}\right).$

(The top part has been left alone and the bottom part has been partially computed).

Invariant 2: $\left(\begin{array}{c} B_T \\ \hline B_B \end{array}\right) = \left(\begin{array}{c} \widehat{B}_T \\ \hline L_{BL}\widehat{B}_T + L_{BR}\widehat{B}_B \end{array}\right).$

(The top part has been left alone and the bottom part has been completely computed).

1.3.2 PME 2

To derive the second PME, partition

$$B \rightarrow \left(\begin{array}{c|c} B_L & B_R \end{array}\right)$$

and do not partition L. Substituting these into the postcondition yields

$$\left(\begin{array}{c|c} B_L & B_R \end{array}\right) = L\left(\begin{array}{c|c} \widehat{B}_L & \widehat{B}_R \end{array}\right)$$

or, equivalently,

$$\left(\begin{array}{c|c} B_L & B_R \end{array}\right) = \left(\begin{array}{c|c} L\widehat{B}_L & L\widehat{B}_R \end{array}\right),$$

which we refer to as the second PME.

From this, we can choose two more loop invariants:

Invariant 3: $\left(\begin{array}{c|c} B_L & B_R \end{array}\right) = \left(\begin{array}{c|c} L\widehat{B}_L & \widehat{B}_R \end{array}\right).$

(The left part has been completely finished and the right part has been left untouched).

Invariant 4: $\left(\begin{array}{c|c} B_L & B_R \end{array}\right) = \left(\begin{array}{c|c} \widehat{B}_L & L\widehat{B}_R \end{array}\right).$

(The left part has been completely finished and the right part has been left untouched).

1.3.3 Notes

How do I decide to partition the matrices in the postcondition?

- Pick a matrix (operand), any matrix.
- If that matrix has
  - a triangular structure (in storage), then you want to either partition is into four quadrants, or not at all. Symmetric matrices and triangular matrices have a triangular structure (in storage).
  - no particular structure, then you partition it vertically (left-right), horizontally (top-bottom), or not at all.
- Next, partition the other matrices similarly, but conformally (meaning the resulting multiplications with the parts are legal).

Take our problem here: $B := LB$. Start by partitioning L in to quadrants:

$$B = \left(\begin{array}{c|c} L_{TL} & 0 \\ \hline L_{BL} & L_{BR} \end{array}\right) \widehat{B}.$$

Now, the way partitioned matrix multiplication works, this doesn't make sense:

$$B = \underbrace{\left(\begin{array}{c|c} L_{TL} & 0 \\ \hline L_{BL} & L_{BR} \end{array}\right) \widehat{B}.}_{\left(\begin{array}{c} L_{TL} \times \text{something} + 0 \times \text{something} \\ \hline L_{BL} \times \text{something} + L_{BR} \times \text{something} \end{array}\right)} \quad .$$

So, we need to also partition B into a top part and a bottom part:

$$\left(\begin{array}{c} B_T \\ \hline B_B \end{array}\right) = \underbrace{\left(\begin{array}{c|c} L_{TL} & 0 \\ \hline L_{BL} & L_{BR} \end{array}\right) \left(\begin{array}{c} \widehat{B}_T \\ \hline \widehat{B}_B \end{array}\right).}_{\left(\begin{array}{c} L_{TL}\widehat{B}_T + 0 \times \widehat{B}_B \\ \hline L_{BL}\widehat{B}_T + L_{BR}\widehat{B}_B \end{array}\right)}$$

Alternatively, what if you don't partition L? You have to partition *something* so let's try partitioning B:

$$\left(\begin{array}{c} B_T \\ \hline B_B \end{array}\right) = L \left(\begin{array}{c} \widehat{B}_T \\ \hline \widehat{B}_B \end{array}\right).$$

But that doesn't work... Instead

$$\left(\begin{array}{c|c} B_L & B_R \end{array}\right) = L \left(\begin{array}{c|c} \widehat{B}_L & \widehat{B}_R \end{array}\right) = \left(\begin{array}{c|c} L\widehat{B}_L & L\widehat{B}_R \end{array}\right)$$

works just fine.

1.4 Deriving all unblocked algorithms

The below table summarizes all loop invariants, with links to all files related to this operation. The worksheet and code skeletons were genered using the ☛ Spark webpage.

	Invariant	Derivations	Implementations
1	$\left(\frac{B_T}{B_B}\right) = \left(\frac{\widehat{B}_T}{L_{BR}\widehat{B}_B}\right)$	PDF	trmm_llnn_unb_var1.mlx trmm_llnn_unb_var1.c
2	$\left(\frac{B_T}{B_B}\right) = \left(\frac{\widehat{B}_T}{L_{BL}B_T + L_{BR}\widehat{B}_B}\right)$	PDF	trmm_llnn_unb_var2.mlx trmm_llnn_unb_var2.c
3	$\left(B_L \mid B_R \right) = \left(L\widehat{B}_L \mid \widehat{B}_R \right)$	PDF	trmm_llnn_unb_var3.mlx trmm_llnn_unb_var3.c
4	$\left(B_L \mid B_R \right) = \left(\widehat{B}_L \mid L\widehat{B}_R \right)$	PDF	trmm_llnn_unb_var4.mlx trmm_llnn_unb_var4.c

1.5 Deriving all blocked algorithms

The below table summarizes all loop invariants, with links to all files related to this operation. The worksheet and code skeletons were genered using the ☛ Spark webpage.

	Invariant	Derivations	Implementations
1	$\left(\frac{B_T}{B_B}\right) = \left(\frac{\widehat{B}_T}{L_{BR}\widehat{B}_B}\right)$	PDF	trmm_llnn_blk_var1.mlx trmm_llnn_blk_var1.c
2	$\left(\frac{B_T}{B_B}\right) = \left(\frac{\widehat{B}_T}{L_{BL}B_T + L_{BR}\widehat{B}_B}\right)$	PDF	trmm_llnn_blk_var2.mlx trmm_llnn_blk_var2.c
3	$\left(B_L \mid B_R \right) = \left(L\widehat{B}_L \mid \widehat{B}_R \right)$	PDF	trmm_llnn_blk_var3.mlx trmm_llnn_blk_var3.c
4	$\left(B_L \mid B_R \right) = \left(\widehat{B}_L \mid L\widehat{B}_R \right)$	PDF	trmm_llnn_blk_var4.mlx trmm_llnn_blk_var4.c