TACC Technical Report TR-09-06

Formal correctness proof of mechanically derived CG methods

Paolo Bientinesi*, Victor Eijkhout†, Maggie Myers‡, Robert van de Geijn‡

* RWTH Aachen, Germany
† Texas Advanced Computing Center, The University of Texas at Austin
‡ Computer Science Department, The University of Texas at Austin

1 Introduction

The FLAME system [6, 10] has long been successfully applied to deriving dense matrix algorithms. In an earlier report [5], we showed that it can also be successfully applied to deriving Krylov methods. However, we left unspecified the search strategy that yielded the operations of the inner loop of a FLAME 'worksheet'. We will now demonstrate that these steps can be derived with a formal correctness proof through Hoare triples.

FLAME uses a 'worksheet', a fixed sequence of operations and predicates, to construct its derivations of algorithms. The worksheet contains an iteration loop that describes the subsequent construction of parts of the algorithm output. In the case of Krylov methods, this iteration loop constructs residuals, search directions, and the various scalar constants involved. The crucial part of the worksheet is the 'update' step, that computes these entities. If this update maintains the correctness of a loop-invariant predicate, the worksheet guarantees the correctness of the overall algorithm.

Since the update step in worksheet for Krylov methods can be fairly elaborate, its correctness proof warrants scrutiny. Also, in our earlier work the exact nature of the search algorithm that would construct this update step was left undetermined. In this report we show that we can kill two birds with one stone: a search process that seeks to satisfy the predicate that holds at the end of the update, is in fact enough to derive the the actual steps.

In this report we will not give a further introduction to FLAME, or its application to deriving Krylov methods; for this we refer to our earlier report. Instead, we will concentrate on the construction and correctness proof of the update.

2 Recapitulation

Using a block formalism for iterative methods that goes back to Householder [8], we describe Krylov methods by coupled recurrences

$$APD = R(I - J), \qquad P(I + U) = R$$

where R is the sequence of residuals, P the sequence of search directions, D a diagonal matrix, U strictly upper triangular, and J a matrix with the value 1 on the first lower subdiagonal: $J_{ij} = \delta_{i,j+1}$. For the Conjugate Gradients (CG) method, we add the condition that $R^t R$ be diagonal. (For a more thorough justification of these equations, see our earlier report [5].)

As a first step in a constructive process for deriving the quantities P, R, D, U in these equations, we formulate these relations as a Partitioned Matrix Expression (PME).

Our PME contains four equations:

1. The computation of new residuals;
2. computation of new search direction;
3. orthogonality of residuals;
4. semi-orthogonality of search directions.

In 3×3 form:

$$
\begin{cases}
\left(\begin{array}{c|c|c} AP_LD_L & Ap_Md_M & AP_{\_R}D_R \end{array}\right) = \left(\begin{array}{c|c|c} R_L & r_M & R_R \end{array}\right) \left(\begin{array}{c|c|c} I_{TL} - J_{TL} & 0 & 0 \\ \hline -e_r^t & 1 & 0 \\ \hline 0 & e_0 & I_{BR} - J_{BR} \end{array}\right) \\
\left(\begin{array}{c|c|c} P_L & p_M & P_R \end{array}\right) \left(\begin{array}{c|c|c} I_{TL} + U_{TL} & u_{TM} & u_{TR} \\ \hline 0 & 1 & u_{MR} \\ \hline 0 & 0 & I_{BR} + U_{BR} \end{array}\right) = \left(\begin{array}{c|c|c} R_L & r_M & R_R \end{array}\right) \\
\left(\begin{array}{c} R_L^t \\ \hline r_M^t \\ \hline R_R^t \end{array}\right) \left(\begin{array}{c|c|c} R_L & r_M & R_R \end{array}\right) = \left(\begin{array}{c|c|c} \Omega_{TL} & 0 & 0 \\ \hline 0 & \omega_{MM} & 0 \\ \hline 0 & 0 & \Omega_{BR} \end{array}\right) \\
\left(\begin{array}{c|c|c} P_L^t AP_L & P_L^t Ap_M & P_L^t AP_R \\ \hline p_M^t AP_L & p_M^t Ap_M & p_M^t AP_R \\ \hline P_R^t AP_L & P_R^t Ap_M & P_R^t AP_R \end{array}\right) = \left(\begin{array}{c|c|c} \star & 0 & 0 \\ \hline \star & \star & 0 \\ \hline \star & \star & \star \end{array}\right).
\end{cases}
\tag{1}
$$

where $\star$ indicates that the exact value is not of interest.

There are several possible invariants, as shown in our earlier work. In sections 3.1 and 3.2 we limit ourselves to the following invariant; in section 3.3 we will consider another possibility:

$$
\begin{cases}
\left(\begin{array}{c} AP_LD_L \end{array}\right) = \left(\begin{array}{cc} R_L & r_M \end{array}\right) \left(\begin{array}{c} J_{TL} \\ j_{ML}^t \end{array}\right) \\
\left(\begin{array}{cc} P_L & p_M \end{array}\right) \left(\begin{array}{cc} U_{TL} & u_{TM} \\ 0 & 1 \end{array}\right) = \left(\begin{array}{cc} R_L & r_M \end{array}\right) \\
\left(\begin{array}{cc} R_L & r_M \end{array}\right)^t \left(\begin{array}{cc} R_L & r_M \end{array}\right) = \begin{pmatrix} \Omega_{TL} & 0 \\ 0 & \omega_{MM} \end{pmatrix} \\
\begin{pmatrix} P_L^t AP_L & P_L^t Ap_M \\ p_M^t AP_L & p_M^t Ap_M \end{pmatrix} = \begin{pmatrix} P_L^t AP_L & 0 \\ p_M^t AP_L & p_M^t Ap_M \end{pmatrix}
\end{cases}
\tag{2}
$$

Since our PME is on 3×3 form, the update step will be on 4×4 matrices. The equations before the FLAME update step (the 'before equations') then become:

$$\begin{cases} AP_0D_0 = \begin{pmatrix} R_0 & r_1 \end{pmatrix} \begin{pmatrix} J_{00} \\ j_{10}^t \end{pmatrix} & \begin{pmatrix} P_0 & p_1 \end{pmatrix} \begin{pmatrix} U_{00} & u_{01} \\ & 1 \end{pmatrix} = \begin{pmatrix} R_0 & r_1 \end{pmatrix}, \\ \begin{pmatrix} R_0^t \\ r_1^t \end{pmatrix} \begin{pmatrix} R_0 & r_1 \end{pmatrix} = \begin{pmatrix} \Omega_0 & 0 \\ 0 & \omega_1 \end{pmatrix} & \begin{pmatrix} P_0^tAP_0 & P_0^tAp_1 \\ p_1^tAP_0 & p_1^tAp_1 \end{pmatrix} = \begin{pmatrix} P_0^tAP_0 & 0 \\ p_1^tAP_0 & p_1^tAp_1 \end{pmatrix} \end{cases} \tag{3}$$

All quantities in these equations are known.

The equations after the update (the 'after equations'), additionally, have

$$\begin{array}{ll} 1 & Ap_1d_1 = r_1 - r_2 \\ 2 & P_0u_{02} + p_1u_{12} + p_2 = r_2 \\ 3a & R_0^tr_2 = 0 \\ 3b & r_1^tr_2 = 0 \\ 4a & P_0^tAp_2 = 0 \\ 4b & p_1^tAp_2 = 0 \\ 5 & \omega_2 = {r_2}^t r_2 \end{array}$$

with the unknowns[1] are p_2, r_2, u_{02}, u_{12}, d_1, ω_2. Note that equations 1 and 2 contain two or more unknowns each.

The derivation, both in a traditional sense and in our earlier, FLAME-based, effort, proceeds as follows.

1. Multiplying the first equation left by r_1^t gives

   $$r_1^tr_2 = r_1^tr_1 - r_1^tAp_1$$

   where we note that $r_1^tr_2 = 0$ is the desired after equation 3b. Therefore, making the choices

   $$\begin{cases} d_1 \leftarrow \frac{r_1^tr_1}{r_1^tAp_1} \\ r_2 \leftarrow r_1 - Ap_1d_1 \end{cases}$$

   causes after equations 1 and 3b to be satisfied.
2. Equation 3a is now also seen to be satisfied.

1. We will use colours to indicate quantities of interest. Quantities still to be computed are represented in red; having been computed they are rendered in green.

3. Some formula manipulation shows that

$$P_0^t A r_2 = P_0 A P_0 u_{02} + P_0^t A p_2$$

is a consequence of the after equations, and we note that $P_0^t A p_2 = 0$ is after equation 4a. This leads to a sufficient definition of u_{02} (provided we also satisfy equation 2, but note that we do not have u_{12} yet).

4. More formula manipulation gives a sufficient value for u_{12}, again with the proviso of equation 2 being satisfied.

5. We conclude that the choices

$$\begin{cases} u_{02} \leftarrow \dots \\ u_{12} \leftarrow \dots \\ p_2 \leftarrow r_2 - P_0 u_{02} - p_1 u_{12} \end{cases}$$

will make equations 2, 4a, 4b satisfied.

In the next section we will give a formal correctness proof of this computation.

3 Proved correct CG

We will give formal correctness proofs of algorithms for the CG method for both symmetric and nonsymmetric systems.

3.1 Correctness proof, nonsymmetric case

Let us consider the first step in the above derivation. We want to have a predicate satisfied:

$$P_2 : \quad \{R_0^t r_2 = 0 \wedge r_1^t r_2 = 0 \wedge A p_1 d_1 = r_1 - r_2\}$$

We decide to arrive at this by assuming that only r_2 is left to be computed. In order to find the actual expression for r_2, we consider the statement and weakest predicate

$$\begin{array}{ll} P_1 : & \{R_0^t x = 0 \wedge r_1^t x = 0 \wedge A p_1 d_1 = r_1 - x\} \\ S_2 : & r_2 \leftarrow x \\ P_2 : & \{r_1^t r_2 = 0 \wedge A p_1 d_1 = r_1 - r_2\} \end{array}$$

The second clause in P_1 suggest that we take $x = r_1 - A p_1 d_1$, so we arrive at the statement

$$S_2 : \quad r_2 \leftarrow r_1 - A p_1 d_1$$

and the weakest precondition that makes P_2 satisfied under assignment S_2:

$$P_1: \quad \{R_0^t(r_1 - Ap_1d_1) = 0 \wedge r_1^t(r_1 - Ap_1d_1) = 0\}$$

Observing that the statements $R_0^t r_1 = 0$ and $R_0^t Ap_1 = 0$ follow from the 'before' equations, the precondition reduces to

$$P_1: \quad \{r_1^t(r_1 - Ap_1d_1) = 0\}$$

We summarize the steps thus far:

$$\begin{array}{ll} P_1: & \{r_1^t(r_1 - Ap_1d_1) = 0\} \\ S_2: & r_2 \leftarrow r_1 - Ap_1d_1 \\ P_2: & \{R_0^t r_2 = 0 \wedge r_1^t r_2 = 0 \wedge Ap_1d_1 = r_1 - r_2\} \end{array}$$

From now on we will dispense with the explicit introduction of x in deriving the desired statement and its weakest precondition, since the expression taken is mostly obvious.

Continuing, in P_1, only d_1 is unknown, so we introduce the statement S_1:

$$\begin{array}{ll} S_1: & d_1 \leftarrow r_1^t r_1 / r_1^t Ap_1 \\ P_1: & \{r_1^t(r_1 - Ap_1d_1) = 0\} \\ S_2: & r_2 \leftarrow r_1 - Ap_1d_1 \\ P_2: & \{r_1^t r_2 = 0 \wedge Ap_1d_1 = r_1 - r_2\} \end{array} \tag{4}$$

The weakest precondition that makes P_1 true after assignment S_1 is the empty predicate, so we now have the following derivation:

$$\begin{array}{ll} P_0: & \{T\} \\ S_1: & d_1 \leftarrow r_1^t r_1 / r_1^t Ap_1 \\ P_1: & \{r_1^t(r_1 - Ap_1d_1) = 0\} \\ S_2: & r_2 \leftarrow r_1 - Ap_1d_1 \\ P_2: & \{r_1^t r_2 = 0 \wedge Ap_1d_1 = r_1 - r_2\} \end{array}$$

Having found the derivation for r_2 and d_1, we concentrate on p_2 and the u_{*2} coefficients. Ultimately, the following predicate needs to be satisfied:

$$P_5: \quad \{r_2 = P_0u_{02} + p_1u_{12} + p_2 \wedge P_0^t Ap_2 = 0 \wedge p_1^t Ap_2 = 0\}$$

We assume that, in the final step, everything but p_2 is known, and we arrive at P_5 by computing p_2:

$$\begin{array}{ll} S_5: & p_2 \leftarrow r_2 - P_0u_{02} + p_1u_{12} \\ P_5: & \{r_2 = P_0u_{02} + p_1u_{12} + p_2 \wedge P_0^t Ap_2 = 0 \wedge p_1^t Ap_2 = 0\} \end{array}$$

Textual substitution tells us that the weakest precondition to make P_5 true under S_5 is:

$$P_4: \quad \{P_0^t A(r_2 - P_0 u_{02} + p_1 u_{12}) = 0 \wedge p_1^t A(r_2 - P_0 u_{02} + p_1 u_{12}) = 0\}$$

which, observing that $P_0^t A p_1 = 0$, reduces to

$$\begin{array}{ll} P_4: & \{P_0^t A(r_2 - P_0 u_{02}) = 0 \wedge p_1^t A(r_2 - P_0 u_{02} + p_1 u_{12}) = 0\} \\ S_5: & p_2 \leftarrow r_2 - P_0 u_{02} + p_1 u_{12} \\ P_5: & \{r_2 = P_0 u_{02} + p_1 u_{12} + p_2 \wedge P_0^t A p_2 = 0 \wedge p_1^t A p_2 = 0\} \end{array}$$

We compute u_{12} to satisfy the second clause:

$$\begin{array}{ll} S_4: & u_{12} \leftarrow (p_1^t A p_1)^{-1}(p_1^t A r_2 - p_1^t A P_0 u_{02}) \\ P_4: & \{P_0^t A(r_2 - P_0 u_{02}) = 0 \wedge p_1^t A(r_2 - P_0 u_{02} + p_1 u_{12}) = 0\} \\ S_5: & p_2 \leftarrow r_2 - P_0 u_{02} + p_1 u_{12} \\ P_5: & \{r_2 = P_0 u_{02} + p_1 u_{12} + p_2 \wedge P_0^t A p_2 = 0 \wedge p_1^t A p_2 = 0\} \end{array}$$

The weakest precondition to satisfy P_4 after S_4 is:

$$\begin{array}{ll} P_3: & \{P_0^t A(r_2 - P_0 u_{02}) = 0 \\ S_4: & u_{12} \leftarrow (p_1^t A p_1)^{-1}(p_1^t A r_2 - p_1^t A P_0 u_{02}) \\ P_4: & \{P_0^t A(r_2 - P_0 u_{02}) = 0 \wedge p_1^t A(r_2 - P_0 u_{02} + p_1 u_{12}) = 0\} \\ S_5: & p_2 \leftarrow r_2 - P_0 u_{02} + p_1 u_{12} \\ P_5: & \{r_2 = P_0 u_{02} + p_1 u_{12} + p_2 \wedge P_0^t A p_2 = 0 \wedge p_1^t A p_2 = 0\} \end{array}$$

and if P_0 is of size $n \times k$, we see that u_{02} is now the solution of a $k \times k$ linear system:

$$\begin{array}{ll} S_3: & u_{02} = (P_0^t A P_0)^{-1} P_0^t A r_2 \\ P_3: & \{P_0^t A(r_2 - P_0 u_{02}) = 0 \\ S_4: & u_{12} \leftarrow (p_1^t A p_1)^{-1}(p_1^t A r_2 - p_1^t A P_0 u_{02}) \\ P_4: & \{P_0^t A(r_2 - P_0 u_{02}) = 0 \wedge p_1^t A(r_2 - P_0 u_{02} + p_1 u_{12}) = 0\} \\ S_5: & p_2 \leftarrow r_2 - P_0 u_{02} + p_1 u_{12} \\ P_5: & \{r_2 = P_0 u_{02} + p_1 u_{12} + p_2 \wedge P_0^t A p_2 = 0 \wedge p_1^t A p_2 = 0\} \end{array}$$

3.2 Correctness proof, symmetric case

We now derive the classic CG method [7], which is based on coupled two-term recurrences. The difference between this method and the one just derived for the general, nonsymmetric, case, is that the search direction p_2 gets updated only from the immediately previous direction p_1, rather than from all previous directions p_1, P_0. We will of course derive this fact in our system.

In the symmetric case, the derivation of r_2 and d_1 is exactly as before, so we take the following steps for given:

$$\begin{array}{ll} P_0: & \{T\} \\ S_1: & d_1 \leftarrow r_1^t r_1 / r_1^t A p_1 \\ P_1: & \{r_1^t (r_1 - A p_1 d_1) = 0\} \\ S_2: & r_2 \leftarrow r_1 - A p_1 d_1 \\ P_2: & \{r_1^t r_2 = 0 \wedge A p_1 d_1 = r_1 - r_2\} \end{array}$$

However, before we can proceed to the u_{*2} coefficients, we need to perform some derivations at the PME level. (The derivations are given here in unpartitioned form; we actually perform them on the 3×3 PME.)

- Take the equation $APD = R(I - J)$ and multiply left by R^t:

  $$R^t AP = R^t R(I - J)D^{-1}$$

  from which we conclude that $R^t AP$ is lower bidiagonal.
- Substitute $R = P(I + U)$:

  $$P^t AP = (I + U)^{-t} R^t R(I - J)D^{-1}$$

  so $P^t AP$ is lower triangular. However, if A is symmetric, $P^t AP$ also is, so it is in fact diagonal.
- Now go back to

  $$(I + U)^t P^t AP = R^t R(I - J)D^{-1}$$

  and, since the right hand side is lower bidiagonal, the left hand side is too, and therefore $I + U$ is upper bidiagonal.

We conclude that, in the column u_{*2} to be computed, $u_{02} = 0$, and only u_{12} remains to be computed.

Now our predicate becomes

$$P_4: \quad \{r_2 = p_1 u_{12} + p_2 \wedge P_0^t A p_2 = 0 \wedge p_1^t A p_2 = 0\}$$

We assume that everything but p_2 is known, and we arrive at P_4 by computing p_2:

$$\begin{array}{ll} S_4: & p_2 \leftarrow r_2 - p_1 u_{12} \\ P_4: & \{r_2 = p_1 u_{12} + p_2 \wedge P_0^t A p_2 = 0 \wedge p_1^t A p_2 = 0\} \end{array}$$

Textual substitution tells us that the weakest precondition to make P_4 true under S_4 is:

$$\begin{array}{ll} P_3: & \{P_0^t A(r_2 - p_1 u_{12}) = 0 \wedge p_1^t A(r_2 - p_1 u_{12}) = 0\} \\ S_4: & p_2 \leftarrow r_2 - p_1 u_{12} \\ P_4: & \{r_2 = p_1 u_{12} + p_2 \wedge P_0^t A p_2 = 0 \wedge p_1^t A p_2 = 0\} \end{array}$$

The clause $P_0^t A(r_2 - p_1 u_{12}) = 0$ is already satisfied since both $P_0^t A r_2 = 0$ and $P_0^t A p_1 = 0$, so we are left with needing to satisfy the clause $p_1^t A(r_2 - p_1 u_{12}) = 0$, which we do by the statement S_3:

$$
\begin{array}{ll}
S_3: & u_{12} \leftarrow (p_1^t A p_1)^{-1} p_1^t A r_2 \\
P_3: & \{P_0^t A(r_2 - p_1 u_{12}) = 0 \wedge p_1^t A(r_2 - p_1 u_{12}) = 0\} \\
S_4: & p_2 \leftarrow r_2 - p_1 u_{12} \\
P_4: & \{r_2 = p_1 u_{12} + p_2 \wedge P_0^t A p_2 = 0 \wedge p_1^t A p_2 = 0\}
\end{array}
$$

It remains to remark that

$$
\frac{p_1^t A r_2}{p_1^t A p_1} = \frac{d_1^{-1} d_1 p_1^t A r_2}{d_1^{-1} d_1 p_1^t A p_1} = \frac{(r_1 - r_2)^t r_2}{(r_1 - r_2)^t p_1} = \frac{-r_2^t r_2}{r_1^t p_1} = -\frac{r_2^t r_2}{r_1^t r_1}
$$

so we have indeed derived the classic CG algorithm here.

3.3 Correctness proof, second invariant for the nonsymmetric case

In our report [5], we derived from the PME of equation (1) two different invariants. Both are equivalent in the nonsymmetric case, but the second invariant is not able to express the two-term recurrence for the search directions in the symmetric case, leading to a wasteful algorithm. Here we will give a correctness proof of the update step of this second invariant.

The PME invariant differs by including one fewer column for the $R = P(I + U)$ equation:

$$
\left(\begin{array}{c|c|c} AP_L D_L & Ap_M d_M & AP_R D_R \end{array} \right) = \left(\begin{array}{c|c|c} R_L & r_M & R_R \end{array} \right) \left(\begin{array}{c|c|c} J_{TL} & 0 & 0 \\ \hline j_{ML}^t & 1 & 0 \\ \hline 0 & j_{MR} & J_{BR} \end{array} \right)
$$

and

$$
\left(\begin{array}{c|c|c} P_L & p_M & P_R \end{array} \right) \left(\begin{array}{c|c|c} U_{TL} & u_{TM} & u_{TR} \\ \hline 0 & 1 & u_{MR} \\ \hline 0 & 0 & U_{BR} \end{array} \right) = \left(\begin{array}{c|c|c} R_L & r_M & R_R \end{array} \right)
$$

given a before equation

$$
A \begin{pmatrix} P_0 \end{pmatrix} \begin{pmatrix} D_0 \end{pmatrix} = \begin{pmatrix} R_0 & r_1 \end{pmatrix} \begin{pmatrix} J_{00} \\ j_{10}^t \end{pmatrix}, \qquad \begin{pmatrix} P_0 \end{pmatrix} \begin{pmatrix} U_{00} \end{pmatrix} = \begin{pmatrix} R_0 \end{pmatrix}
$$

and after equations

$$A \begin{pmatrix} P_0 & p_1 \end{pmatrix} \begin{pmatrix} D_0 & \\ & d_1 \end{pmatrix} = \begin{pmatrix} R_0 & r_1 & r_2 \end{pmatrix} \begin{pmatrix} J_{00} & j_{01} & \\ j_{10}^t & 1 & -1 \\ & & 1 \end{pmatrix}$$

$$\begin{pmatrix} P_0 & p_1 \end{pmatrix} \begin{pmatrix} U_{00} & u_{01} \\ & 1 \end{pmatrix} = \begin{pmatrix} R_0 & r_1 \end{pmatrix}$$

The update needs to satisfy the following set of equations:

$$\begin{cases} Ap_1d_1 = r_1 - r_2 \\ P_0u_{01} + p_1 = r_1 \\ R_0^t r_2 = 0 \\ r_1^t r_2 = 0 \\ P_0^t Ap_1 = 0 \end{cases}$$

We will now given an accelerated account of the derivation of this predicate; we will underline the clauses that we are targeting.

Since this invariant is, in a way, equivalent to moving the p update from the end of an iteration to the start, we now end by computing r_2:

$$\begin{array}{ll} S_4: & r_2 \leftarrow r_1 - d_1 p_1^t Ap_1 \\ P_4: & \{\underline{Ap_1d_1 = r_1 - r_2} \wedge P_0u_{01} + p_1 = r_1 \wedge R_0^t r_2 = 0 \wedge r_1^t r_2 = 0 \wedge P_0^t Ap_1 = 0\} \end{array}$$

In the step before this we computed d_1:

$$\begin{array}{ll} S_3: & d_1 \leftarrow r_1^t r_1 / r_1^t Ap_1 \\ P_3: & \{P_0u_{01} + p_1 = r_1 \wedge R_0^t(r_1 - Ap_1d_1) = 0 \wedge \underline{r_1^t(r_1 - Ap_1d_1) = 0} \wedge P_0^t Ap_1 = 0\} \\ S_4: & r_2 \leftarrow r_1 - d_1 p_1^t Ap_1 \\ P_4: & \{Ap_1d_1 = r_1 - r_2 \wedge P_0u_{01} + p_1 = r_1 \wedge R_0^t r_2 = 0 \wedge r_1^t r_2 = 0 \wedge P_0^t Ap_1 = 0\} \end{array}$$

Before this we computed p_1:

$$\begin{array}{ll} S_2: & p_1 \leftarrow r_1 - P_0u_{01} \\ P_2: & \{\underline{P_0u_{01} + p_1 = r_1} \wedge P_0^t Ap_1 = 0\} \\ S_3: & d_1 \leftarrow r_1^t r_1 / r_1^t Ap_1 \\ P_3: & \{P_0u_{01} + p_1 = r_1 \wedge R_0^t(r_1 - Ap_1d_1) = 0 \wedge \underline{r_1^t(r_1 - Ap_1d_1) = 0} \wedge P_0^t Ap_1 = 0\} \\ S_4: & r_2 \leftarrow r_1 - d_1 p_1^t Ap_1 \\ P_4: & \{Ap_1d_1 = r_1 - r_2 \wedge P_0u_{01} + p_1 = r_1 \wedge R_0^t r_2 = 0 \wedge r_1^t r_2 = 0 \wedge P_0^t Ap_1 = 0\} \end{array}$$

Finally, we find the initial computation of u_{01}:

$$
\begin{array}{ll}
P_0: & \{T\} \\
S_1: & u_{01} \leftarrow (P_0^t A P_0)^{-1} P_0^t A r_1 \\
P_1: & \{\underline{P_0^t A r_1 - P_0 u_{01} = 0}\} \\
S_2: & p_1 \leftarrow r_1 - P_0 u_{01} \\
P_2: & \{\underline{P_0 u_{01} + p_1 = r_1} \wedge P_0^t A p_1 = 0\} \\
S_3: & d_1 \leftarrow r_1^t r_1 / r_1^t A p_1 \\
P_3: & \{P_0 u_{01} + p_1 = r_1 \wedge R_0^t (r_1 - A p_1 d_1) = 0 \wedge \underline{r_1^t (r_1 - A p_1 d_1) = 0} \wedge P_0^t A p_1 = 0\} \\
S_4: & r_2 \leftarrow r_1 - d_1 p_1^t A p_1 \\
P_4: & \{A p_1 d_1 = r_1 - r_2 \wedge P_0 u_{01} + p_1 = r_1 \wedge R_0^t r_2 = 0 \wedge r_1^t r_2 = 0 \wedge P_0^t A p_1 = 0\}
\end{array}
\tag{5}
$$

The computation of u_{01} involves solving a system with $P_0^t A P_0$. This is not prohibitively expensive since the system is lower triangular, and of size the number of iterations.

4 Computational variants of CG

We will now address the possibility of deriving mathematically equivalent, but possibly computationally differently behaved, variants of the CG method.

4.1 'Essentially equivalent' methods

In equation (4) we derived a value $r_1^t r_1 / r_1^t A p_1$ for the quantity d_1, while the expression typically given is

$$d_1 = \frac{r_1^t r_1}{p_1^t A p_1}.$$

While it is easy to derive the statement $r_1^t A p_1 = p_1^t A p_1$, we can not use this to declare the corresponding two variants of CG numerically equivalent, since one expression can be more numerically stable than the other. Therefore, it would be preferable if our derivation stategy could derive both, after which further analysis or numerical experiments would pronounce a preference for either.

For this, we observe that we can substitute the 'before' equation

$$\begin{pmatrix} R_0 & r_1 \end{pmatrix} = \begin{pmatrix} P_0 & p_1 \end{pmatrix} \begin{pmatrix} U_{00} & u_{01} \\ 0 & 1 \end{pmatrix}$$

in

$$\begin{pmatrix} P_0^t \\ p_1^t \end{pmatrix} \begin{pmatrix} AP_0 & Ap_1 \end{pmatrix},$$

giving

$$\begin{pmatrix} R_0^t AP_0 & \\ r_1^t AP_0 & r_1^t Ap_1 \end{pmatrix} = \begin{pmatrix} U_{00}^t & \\ u_{10}^t & 1 \end{pmatrix} \begin{pmatrix} P_0^t AP_0 & \\ p_1^t AP_0 & p_1^t Ap_1 \end{pmatrix} \tag{6}$$

from which we find $r_1^t Ap_1 = p_1^t Ap_1$ by considering the $(1,1)$ position, and this identity holds both for the symmetric and nonsymmetric methods. The correctness proof would be able to make this substitution if the components of equation (6) are among the known facts. The derivation step of equation (4) then becomes

$$\begin{array}{ll} P_0: & \{r_1^t Ap_1 = p_1^t Ap_1\} \\ S_1: & d_1 \leftarrow r_1^t r_1 / p_1^t Ap_1 \\ P_1: & \{r_1^t (r_1 - Ap_1 d_1) = 0\} \end{array}$$

On reconsideration, we see that in equation (5) we should let the predicate P_0 be the sum of all identities that are derivable from the 'before' equations[2].

4.2 True computational variants

The CG method as formulated above has two inner products per iteration. Since an inner product induces a synchronization point in a parallel context, people have formulated methods that compute the same vectors, but that allow grouping of the inner products [1, 2, 3, 4, 9, 11].

One such method for the classical CG algorithm is based on the equation

$$R^t AR = (I+U)^t P^t AP(I+U). \tag{7}$$

Inspired by this, we note that we can derive an extra 'before' equation

$$r_1^t Ar_1 = p_1^t Ap_1 + u_{01}^t P_0^t AP_0 u_{01}. \tag{8}$$

This means that we can compute the quantity $r_1^t Ar_1$ by computing an inner product and derive the $p_1^t Ap_1$ from it using a purely scalar equation. The inner product $r_1^t Ar_1$ can

2. We observe that this collection of facts is limited in a certain sense: only a limited number of substitutions can be made – after all, we only have a finite number of equations, and the only way to increase the number of equations is to incorporate higher powers of A. In classical CG proofs we reason about sequences, so we can expand a large number of terms, until we reach the initial vector in the sequence. Does giving this up limit us? Good question.

be computed simultaneously with $r_1^t r_1$, thereby removing one synchronization point from the algorithm. Using equation (8) and the invariant of section 3.3, we observe that all terms are available at the start of the loop body, but the computation has a number of inner products that is quadratic in the current number of iterations. To reduce this to linear, we can accumulate the matrix of $P_0^t A P_0$ values. Our current framework is not able to derive this cost saving. Possibly this can be realized through compiler optimizations.

However, a better strategy does not rely on deriving extra 'before' equation. Instead, we add equation 7 to the PME as

$$\left(\begin{array}{c|c|c} R_L^t A R_L & R_L^t A r_M & R_L^t A R_R \\ \hline r_M^t A R_L & r_M^t A r_M & r_M^t A R_R \\ \hline R_R^t A R_L & R_R^t A r_M & R_R^t A R_R \end{array}\right) =$$

$$\left(\begin{array}{c|c|c} I_{TL} + U_{TL}^t & 0 & 0 \\ \hline u_{TM}^t & 1 & 0 \\ \hline u_{TR}^t & u_{MR}^t & I_{BR} + U_{BR}^t \end{array}\right) \left(\begin{array}{c|c|c} P_L^t A P_L & P_L^t A p_M & P_L^t A P_R \\ \hline p_M^t A P_L & p_M^t A p_M & p_M^t A P_R \\ \hline P_R^t A P_L & P_R^t A p_M & P_R^t A P_R \end{array}\right) \left(\begin{array}{c|c|c} I_{TL} + U_{TL} & u_{TM} & u_{TR} \\ \hline 0 & 1 & u_{MR} \\ \hline 0 & 0 & I_{BR} + U_{BR} \end{array}\right)$$

This gives us a further 'after' equation

$$\begin{aligned} r_2^t A r_2 &= p_2^t A p_2 + u_{02}^t P_0^t A P_0 u_{02} + u_{12} p_1^t A p_1 u_{12} \\ &= p_2^t A p_2 + u_{12} p_1^t A p_1 u_{12} \end{aligned} \tag{9}$$

Important note. We are introducing the matrix $R^t A R$ into the PME, but only its diagonal needs to be computed. It is not yet clear how this fact should be expressed in our framework.

5 Conclusion and discussion

In this report we have shown how Krylov methods can be derived hand-in-hand with their correctness proof. This builds on our earlier work where we showed that Krylov methods can be derived in the FLAME system. However, FLAME leaves the correctness of the update step unspecified. Here, we have shown that the update step can be derived as a sequence of Hoare triples, thereby guaranteeing its correctness.

A number of research questions remain open. We have given a solution for the problem of computational variants in section 4. However, they may also be arrived at by using different invariants, or they can be derived through post-processing on methods already derived in a FLAME worksheet. It is not yet clear what will be the most natural way of approaching this.

References

[1] A. Chronopoulos and C.W. Gear. s-step iterative methods for symmetric linear systems. *Journal of Computational and Applied Mathematics*, 25:153–168, 1989.

[2] E.F. D'Azevedo, V.L. Eijkhout, and C.H. Romine. Lapack working note 56: Reducing communication costs in the conjugate gradient algorithm on distributed memory multiprocessor. Technical Report CS-93-185, Computer Science Department, University of Tennessee, Knoxville, 1993.

[3] E.F. D'Azevedo, V.L. Eijkhout, and C.H. Romine. A matrix framework for conjugate gradient methods and some variants of cg with less synchronization overhead. In *Proceedings of the Sixth SIAM Conference on Parallel Procesing for Scientific Computing*, pages 644–646, Philadelphia, 1993. SIAM.

[4] J. Demmel, M. Heath, and H. Van der Vorst. Parallel numerical linear algebra. In *Acta Numerica 1993*. Cambridge University Press, Cambridge, 1993.

[5] Victor Eijkhout, Paolo Bientinesi, and Robert van de Geijn. Formal derivation of Krylov methods. Technical Report TR-08-03, Texas Advanced Computing Center, The University of Texas at Austin, 2008.

[6] John A. Gunnels, Fred G. Gustavson, Greg M. Henry, and Robert A. van de Geijn. Flame: Formal linear algebra methods environment. *ACM Transactions on Mathematical Software*, 27(4):422–455, December 2001.

[7] M.R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *Nat. Bur. Stand. J. Res.*, 49:409–436, 1952.

[8] Alston S. Householder. *The theory of matrices in numerical analysis*. Blaisdell Publishing Company, New York, 1964. republished by Dover Publications, New York, 1975.

[9] Gerard Meurant. Multitasking the conjugate gradient method on the CRAY X-MP/48. *Parallel Computing*, 5:267–280, 1987.

[10] Robert A. van de Geijn and Enrique S. Quintana-Ortí. *The Science of Programming Matrix Computations*. `www.lulu.com`, 2008.

[11] L. T. Yand and R. Brent. The improved bicgstab method for large and sparse unsymmetric linear systems on parallel distributed memory architectures. In *Proceedings of the Fifth International Conference on Algorithms and Architectures for Parallel Processing*. IEEE, 2002.