Appendix A: Loop structure of the 4-ext integral transformation

Here, we describe briefly the new integral transformation module, which generates the 4-external integral distribution: Two of the three permutational symmetries of the AO integrals are exploited, thus each unique AO integral is effectively computed twice. The third permutational symmetry within the slower shell pair (MN) is discarded in order to avoid simultaneous storage of all three-quarter transformed integrals on disk. Note that the latter set is considerably larger than the final set of fully transformed 4-external integrals. The third and fourth quarter transformation steps are driven by the sparse atom pair and triple lists, and in each transformation step a fast dense matrix multiply constitutes the computational kernel. The partially transformed integrals of each step are stored in sparse form, i.e., as a sparse list of locally dense integral blocks over shell and atom ranges.

```
DO M=1,NShell
  DO N=1,NShell
    Reset memory for (MN|Rr) blocks (Q1 memory)
    DO R=1,Max(M,R)
      DO S=1,R (S=1,Min(M,N) if R==Max(M,N) )
        Compute integral block (MN|RS)
        Check, if new (MN|Rr) blocks will contribute (prescreening)
        If so, increase Q1 memory pointer
        Q1 step over shell block:
        Q1(MN|Rr) = Q1(MN|Rr) + (MN|RS) * P(S,r)
        Q1(MN|Sr) = Q1(MN|Sr) + (MN|RS) * P(R,r)
      END DO
    END DO
    DO R=1,NShell
      IF (MN|Rr) exists THEN
        -- s: (r,s) in AtomPairLst, prescreening
        Q2(MN|sr) = Q2(MN|sr) + Q1(MN|Rr) * P(R,s)
      END IF
    END DO
```

```
      Check, if new (MN|sr) blocks will contribute (prescreening)
      If so, increase Q2 disk pointer
      Write Q2(MN|sr) to disk in canonical order:
      (for each AtomPair corresponding Q2 integral block,
       with AOs mu,nu fixed to shells M,N)
    END DO
    LOOP over AtomPairLst
      DO N=1,NShell
        IF (MN|sr) exists THEN
          Read Q2(MN|sr) from disk
          -- t: (r,s,t) in AtomTriplesLst, prescreening
          Q3(Mt|sr) = Q3(Mt|sr) + Q2(MN|sr) * P(N,t)
          Write Q3(Mt|sr) to disk
        END IF
      END DO
    END LOOP
    IF (overall Q3(Mt|sr) integrals exceed disk buffer) THEN
      LOOP over AtomTriplesLst
        LOOP over M Shell range
          IF (Q3(Mt|sr) exists THEN
            Read Q3(Mt|sr) from disk
            -- u: (r,s,t,u) in AtomQuadLst, prescreening
            Read Q4(ut|sr) from disk
            Q4(ut|sr) = Q4(ut|sr) + Q3(Mt|sr) * P(M,u)
            Write Q4(ut|sr) to disk
          END IF
        END LOOP
      END LOOP
    END IF
  END DO
  Perform Q4 step analogously for remaining Q3(Mt|sr) integrals
  in disk buffer
```