Chapter 5

Sample Programs

This chapter presents several very simple programming examples in CM Fortran.

- A prime number sieve
- A solver for Laplace's equation

5.1 Prime Number Sieve

Beginning on the following page is a program that finds the prime numbers in a set of numbers. Three versions of the program are presented for comparison: a serial version and two alternative parallel versions.

Primes: Fortran 77 Version

```
      PROGRAM FINDPRIMES
      INTEGER I, J, N
      PARAMETER (N = 999)
      LOGICAL PRIMES(N), CANDID(N)
C
C     Initialization
C
      DO 1 I=1,N
         PRIMES(I) = .FALSE.
         CANDID(I) = .TRUE.
 1    CONTINUE
      CANDID(1) = .FALSE.
C
C     Loop: Find next valid candidate, mark it as prime,
C           invalidate all multiples as candidates, repeat.
C
 2    DO 4 I=1,SQRT(REAL(N))
         IF (CANDID(I)) THEN
            PRIMES(I) = .TRUE.
            DO 3 J=I,N,I
               CANDID(J) = .FALSE.
 3          CONTINUE
         END IF
 4    CONTINUE
C
C     At this point, all valid candidates are prime
C
      DO 5 I=SQRT(REAL(N))+1,N
         PRIMES(I) = CANDID(I)
 5    CONTINUE

C
C     Print results
C


      DO I=1,N
         IF (PRIMES(I)) PRINT *,I
      END DO
      END
```

Primes: First CM Fortran Version

This first parallel version is a straightforward translation of the serial program.

```
      PROGRAM FINDPRIMES_1
      IMPLICIT NONE        ! suppress implicit typing rules
      INTEGER I, N, NEXTPRIME
      PARAMETER (N = 999)
      LOGICAL PRIMES(N), CANDID(N)

C     Initialization
C
      PRIMES = .FALSE.
      CANDID = .TRUE.
      CANDID(1) = .FALSE.
C
C     Loop: Find next valid candidate, mark it as prime,
C           invalidate all multiples as candidates, repeat.
C           Notice support for DO WHILE and END DO.
C
      NEXTPRIME = 2
      DO WHILE ( NEXTPRIME .LE. SQRT( REAL(N) ) )
         PRIMES( NEXTPRIME ) = .TRUE.
         CANDID( NEXTPRIME:N:NEXTPRIME ) = .FALSE.
         NEXTPRIME = MINVAL( [1:N], DIM=1, MASK=CANDID )
      END DO
C
C     At this point, all valid candidates are prime.
C
      PRIMES( NEXTPRIME:N ) = CANDID( NEXTPRIME:N )
C
C     Print results
C
      PRINT *, ƒNumber of primes:ƒ, COUNT(PRIMES)
      DO I=1, N
         IF (PRIMES(I)) PRINT *, I
      END DO
      END
```

Primes: Second CM Fortran Version

This parallel version uses a different approach from the two programs just shown. This algorithm uses more processors than the other parallel version.

```
      PROGRAM FINDPRIMES_2
      IMPLICIT NONE
      INTEGER I, N, NN
      PARAMETER (N = 500)
      INTEGER TEMP1(N,N), TEMP2(N,N)
      LOGICAL CANDID(N,N), PRIMES(N)

      CANDID = .FALSE.
      FORALL (I=1:N)  TEMP1(I,:) = 2*I+1
      FORALL (I=1:N)  TEMP2(:,I) = 2*I+1
C
C     The temporary 2-dimensional arrays are now modulated
C     element by element. If an element in TEMP1 is not a
C     multiple of the corresponding element of TEMP2, or (for
C     the sake of an easily generated argument to the upcoming
C     ALL instrinsic) the element in TEMP1 is greater than or
C     equal to the corresponding element in TEMP2, then the
C     corresponding element in the CANDID array is set.
C
      WHERE ( ((MOD( TEMP1,TEMP2 ) .NE. 0)
     $             .AND. (TEMP1 .GT. TEMP2))
     $             .OR.   (TEMP1 .LE. TEMP2) )
     $       CANDID = .TRUE.

C
C     Perform an AND across the second dimension of CANDID.
      PRIMES = ALL(CANDID, DIM=2)
C
C     Print results
      PRINT *, ƒNumber of primes:ƒ, COUNT(PRIMES)+1
      PRINT *, 2
      DO I=1,N
         IF (PRIMES(I)) PRINT *, 2*I+1
      END DO
      END
```

5.2 Laplace Solver

This program solves Laplace's equation

$$\nabla^2 f = 0$$

on the unit square (**[0,1]** x **[0,1]**), subject to the boundary condition that $f = 1$ at $y = 1$ and $f = 2$ along the rest of the boundary. This program uses the 5-point Jacobi relaxation method, with f initially set to 0 on the interior.

```
      PROGRAM LAPLACE
      PARAMETER (MAXX=32)
      PARAMETER (MAXY=MAXX)
      REAL F(MAXX,MAXY),DF(MAXX,MAXY)
      LOGICAL CMASK(MAXX,MAXY)
      REAL RMS_ERROR,MAX_ERROR
      INTEGER ITERATION
C
C     Initialize mask and F for boundaries and interior
C
      CMASK = .FALSE.
      CMASK( 2:MAXX-1, 2:MAXY-1 ) = .TRUE.

      F = 2.
      F( :,MAXY ) = 1.
      WHERE (CMASK) F = 0.

      MAX_ERROR = 1.
      ITERATION = 0
C
C     Iterate until MAX_ERROR < 1.E-3
C
      DO WHILE (MAX_ERROR.GT.1.E-3)
         ITERATION = ITERATION + 1
C
C        Compute DF, the change at each iteration, and update
C
         DF = 0.
         WHERE (CMASK)
            DF = 0.25*(CSHIFT(F,1,1)+CSHIFT(F,1,-1)+
     S         CSHIFT(F,2,1) + CSHIFT(F,2,-1)) - F
            F = F + DF
         END WHERE
```

```
C
C      Compute the RMS and Maximum errors.
C
          RMS_ERROR = SQRT(SUM(DF*DF)/((MAXX-2)*(MAXY-2)))
          MAX_ERROR = MAXVAL(DF,MASK=CMASK)
C
C      See if we should print things out
C
          IF (MOD(ITERATION,10).EQ.0) THEN
              WRITE (6,*) ITERATION,RMS_ERROR,MAX_ERROR
          END IF
       END DO
C
C      Write the final iteration count
C
       WRITE (6,*) ITERATION,RMS_ERROR,MAX_ERROR

       END
```