

---

```
1.  procedure MAT_VECT ( $A, x, y$ )
2.  begin
3.    for  $i := 0$  to  $n - 1$  do
4.    begin
5.       $y[i] := 0;$ 
6.      for  $j := 0$  to  $n - 1$  do
7.         $y[i] := y[i] + A[i, j] \times x[j];$ 
8.      endfor;
9.    end MAT_VECT
```

---

**Algorithm 8.1** A serial algorithm for multiplying an  $n \times n$  matrix  $A$  with an  $n \times 1$  vector  $x$  to yield an  $n \times 1$  product vector  $y$ .

---

```
1. procedure MAT_MULT ( $A, B, C$ )
2. begin
3.   for  $i := 0$  to  $n - 1$  do
4.     for  $j := 0$  to  $n - 1$  do
5.       begin
6.          $C[i, j] := 0$ ;
7.         for  $k := 0$  to  $n - 1$  do
8.            $C[i, j] := C[i, j] + A[i, k] \times B[k, j]$ ;
9.         endfor;
10.    end MAT_MULT
```

---

**Algorithm 8.2** The conventional serial algorithm for multiplication of two  $n \times n$  matrices.

---

```
1.  procedure BLOCK_MAT_MULT ( $A, B, C$ )
2.  begin
3.    for  $i := 0$  to  $q - 1$  do
4.      for  $j := 0$  to  $q - 1$  do
5.        begin
6.          Initialize all elements of  $C_{i,j}$  to zero;
7.          for  $k := 0$  to  $q - 1$  do
8.             $C_{i,j} := C_{i,j} + A_{i,k} \times B_{k,j}$ ;
9.          endfor;
10. end BLOCK_MAT_MULT
```

---

**Algorithm 8.3** The block matrix multiplication algorithm for  $n \times n$  matrices with a block size of  $(n/q) \times (n/q)$ .

---

```

1.  procedure GAUSSIAN_ELIMINATION ( $A, b, y$ )
2.  begin
3.      for  $k := 0$  to  $n - 1$  do           /* Outer loop */
4.      begin
5.          for  $j := k + 1$  to  $n - 1$  do
6.               $A[k, j] := A[k, j]/A[k, k]$ ; /* Division step */
7.               $y[k] := b[k]/A[k, k]$ ;
8.               $A[k, k] := 1$ ;
9.              for  $i := k + 1$  to  $n - 1$  do
10.             begin
11.                 for  $j := k + 1$  to  $n - 1$  do
12.                      $A[i, j] := A[i, j] - A[i, k] \times A[k, j]$ ; /* Elimination step */
13.                      $b[i] := b[i] - A[i, k] \times y[k]$ ;
14.                      $A[i, k] := 0$ ;
15.                 endfor;           /* Line 9 */
16.             endfor;           /* Line 3 */
17.         end GAUSSIAN_ELIMINATION

```

---

**Algorithm 8.4** A serial Gaussian elimination algorithm that converts the system of linear equations  $Ax = b$  to a unit upper-triangular system  $Ux = y$ . The matrix  $U$  occupies the upper-triangular locations of  $A$ . This algorithm assumes that  $A[k, k] \neq 0$  when it is used as a divisor on lines 6 and 7.

---

```
1. procedure BACK_SUBSTITUTION ( $U, x, y$ )
2. begin
3.   for  $k := n - 1$  downto 0 do /* Main loop */
4.     begin
5.        $x[k] := y[k]$ ;
6.       for  $i := k - 1$  downto 0 do
7.          $y[i] := y[i] - x[k] \times U[i, k]$ ;
8.       endfor;
9.     end BACK_SUBSTITUTION
```

---

**Algorithm 8.5** A serial algorithm for back-substitution.  $U$  is an upper-triangular matrix with all entries of the principal diagonal equal to one, and all subdiagonal entries equal to zero.

---

```
1.  procedure CHOLESKY (A)
2.  begin
3.    for k := 0 to n - 1 do
4.      begin
5.         $A[k, k] := \sqrt{A[k, k]}$ ;
6.        for j := k + 1 to n - 1 do
7.           $A[k, j] := A[k, j]/A[k, k]$ ;
8.        for i := k + 1 to n - 1 do
9.          for j := i to n - 1 do
10.            $A[i, j] := A[i, j] - A[k, i] \times A[k, j]$ ;
11.        endfor;      /* Line 3 */
12.  end CHOLESKY
```

---

**Algorithm 8.6** A row-oriented Cholesky factorization algorithm.

---

1. **procedure** MAT\_MULT\_CREW\_PRAM ( $A, B, C, n$ )
2. **begin**
3.     Organize the  $n^2$  processes into a logical mesh of  $n \times n$ ;
4.     **for** each process  $P_{i,j}$  **do**
5.         **begin**
6.              $C[i, j] := 0$ ;
7.             **for**  $k := 0$  **to**  $n - 1$  **do**
8.                  $C[i, j] := C[i, j] + A[i, k] \times B[k, j]$ ;
9.             **endfor**;
10.     **end** MAT\_MULT\_CREW\_PRAM

---

**Algorithm 8.7** An algorithm for multiplying two  $n \times n$  matrices  $A$  and  $B$  on a CREW PRAM, yielding matrix  $C = A \times B$ .

---

```
1. procedure MAT_MULT_EREW_PRAM ( $A, B, C, n$ )
2. begin
3.   Organize the  $n^2$  processes into a logical mesh of  $n \times n$ ;
4.   for each process  $P_{i,j}$  do
5.     begin
6.        $C[i, j] := 0$ ;
7.       for  $k := 0$  to  $n - 1$  do
8.          $C[i, j] := C[i, j] +$ 
            $A[i, (i + j + k) \bmod n] \times B[(i + j + k) \bmod n, j]$ ;
9.       endfor;
10.  end MAT_MULT_EREW_PRAM
```

---

**Algorithm 8.8** An algorithm for multiplying two  $n \times n$  matrices  $A$  and  $B$  on an EREW PRAM, yielding matrix  $C = A \times B$ .