Qüestió 1 (1 punt)

Implementa, mitjançant operacions punt a punt, una funció amb la següent capçalera:

```
void comunica(double x[], int sizes[], double xloc[], int nloc, int root)
```

la qual ha de repartir el vector `x`, que es troba en el procés d'índex `root`, entre tots els processos del programa MPI, de manera que el procés 0 obtinga el primer bloc d'elements, de grandària `sizes[0]`, el procés 1 obtinga el següent bloc, de grandària `sizes[1]`, etc. La longitud del vector `sizes` és igual al número de processos.

Els arguments `x` i `sizes` només són vàlids en el procés `root`. El valor de `nloc`, que pot ser distint en cada procés, indica la grandària del bloc que li correspon al propi procés. Cada procés, incloent el procés `root`, emmagatzemarà en l'array `xloc` el bloc que li correspon.

Per exemple, si `root` val 0 i la resta d'arguments són:

	P_0						P_1	P_2
`x`	1	4	5	8	9	3	-	-
`sizes`			2	1	3		-	-
`nloc`	2						1	3

el contingut final de `xloc` en cada procés ha de ser:

	P_0		P_1	P_2		
`xloc`	1	4	5	8	9	3

Solució:

```
void comunica(double x[], int sizes[], double xloc[], int nloc, int root) {
  int p, rank, i, iproc, k;
  MPI_Comm_size(MPI_COMM_WORLD,&p);
  MPI_Comm_rank(MPI_COMM_WORLD,&rank);
  if (rank==root) {
    k = 0;
    for (iproc=0; iproc<p; iproc++) {
      if (iproc==rank) {
        /* Copia del bloc de x a xloc */
        for (i=0; i<nloc; i++) xloc[i] = x[k+i];
      }
      else
        MPI_Send(&x[k],sizes[iproc],MPI_DOUBLE,iproc,0,MPI_COMM_WORLD);
      k += sizes[iproc];
    }
  }
  else {
    MPI_Recv(xloc,nloc,MPI_DOUBLE,root,0,MPI_COMM_WORLD,MPI_STATUS_IGNORE);
  }
}
```

Qüestió 2 (1.2 punts)

La següent funció multiplica element a element els valors d'una matriu A per una matriu B i per un número real a, a fi d'obtenir una altra matriu C resultant:

```
void producte_elements(double a,double A[M][N],double B[M][N],double C[M][N]) {
   int i,j;
   for (i=0;i<M;i++) {
      for (j=0;j<N;j++) {
         C[i][j]=a*A[i][j]*B[i][j];
      }
```

```
    }
}
```

0.9 p. (a) Paralelitzar la funció mitjançant operacions col·lectives de MPI, repartint les matrius per blocs de files consecutives. S'entendrà que:

- El procés 0 serà el que dispose inicialment del valor de a i de les matrius A i B.
- Al finalitzar la funció, tots els processos hauran de disposar de la matriu C completa.
- El número de files M de les matrius sempre serà múltiple del número de processos emprats.

Solució:

```
void producte_elements(double a,double A[M][N],double B[M][N],double C[M][N]) {
   int i,j,k,np;
   double Alocal[M][N],Blocal[M][N],Clocal[M][N];
   MPI_Comm_size(MPI_COMM_WORLD,&np);
   k=M/np;
   MPI_Bcast(&a,1,MPI_DOUBLE,0,MPI_COMM_WORLD);
   MPI_Scatter(A,k*N,MPI_DOUBLE,Alocal,k*N,MPI_DOUBLE,0,MPI_COMM_WORLD);
   MPI_Scatter(B,k*N,MPI_DOUBLE,Blocal,k*N,MPI_DOUBLE,0,MPI_COMM_WORLD);
   for (i=0;i<k;i++) {
      for (j=0;j<N;j++) {
         Clocal[i][j]=a*Alocal[i][j]*Blocal[i][j];
      }
   }
   MPI_Allgather(Clocal,k*N,MPI_DOUBLE,C,k*N,MPI_DOUBLE,MPI_COMM_WORLD);
}
```

0.3 p. (b) Calcula el cost aritmètic i el cost de les comunicacions de la funció implementada en l'apartat anterior. Indica clarament el cost total corresponent a cadascuna de las operacions col·lectives emprades.

Solució: En primer lloc, el cost aritmètic seria:

$$t_a(M,p) = \sum_{i=0}^{M/p-1} \sum_{j=0}^{N-1} 2 = \sum_{i=0}^{M/p-1} 2N = \frac{2MN}{p} flops$$

En segon lloc, el cost de les comunicacions es correspondria amb:

$$t_c(M,p) = t_{Bcast} + t_{ScatterA} + t_{ScatterB} + t_{Allgather}$$

En detall:

$$\begin{aligned} t_{Bcast} &= (p-1)(t_s + t_w) \\ t_{ScatterA} &= (p-1)(t_s + \tfrac{MN}{p} t_w) \\ t_{ScatterB} &= (p-1)(t_s + \tfrac{MN}{p} t_w) \\ t_{Allgather} &= (p-1)(t_s + \tfrac{MN}{p} t_w) + (p-1)(t_s + MNt_w) \end{aligned}$$

Qüestió 3 (1.3 punts)

El següent programa realitza el càlcul de la norma d'una matriu quadrada de grandària $3N \times 3N$:

```
double norma(double A[N*3][N*3]) {
    int i, j;
    double norm=0.0;

    for (i=0;i<N*3;i++)
      for (j=0;j<N*3;j++)
         norm += A[i][j]*A[i][j];

    norm=sqrt(norm);
    MPI_Type_free(&type);
    return norm;
}
```

1 p.

(a) Implementa una versió paral·lela en MPI per a 9 processos. La funció ha de distribuir la matriu entre els 9 processos usant una estructura bidimensional com la que es mostra en l'exemple. Cada procés rebrà una submatriu quadrada de A de grandària $N \times N$ que emmagatzemarà en una altra matriu també de grandària $N \times N$. S'ha de minimitzar el número de missatges i evitar còpies intermèdies. NOTA: L'exemple es per a una matriu 6×6 però el programa haurà de funcionar per a una matriu qualsevol de grandària $3N \times 3N$. Se suposa que el número de processos serà 9. El resultat final (`norm`) ha de ser correcte en el procés 0.

$$A = \left(\begin{array}{cc|cc|cc} a_{00} & a_{01} & a_{02} & a_{03} & a_{04} & a_{05} \\ a_{10} & a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ \hline a_{20} & a_{21} & a_{22} & a_{23} & a_{24} & a_{25} \\ a_{30} & a_{31} & a_{32} & a_{33} & a_{34} & a_{35} \\ \hline a_{40} & a_{41} & a_{42} & a_{43} & a_{44} & a_{45} \\ a_{50} & a_{51} & a_{52} & a_{53} & a_{54} & a_{55} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} P_0 & P_1 & P_2 \\ \hline P_3 & P_4 & P_5 \\ \hline P_6 & P_7 & P_8 \end{array}\right)$$

Solució:

```
double norma(double A[N*3][N*3]) {
    int rank;
    int i, j, proc;
    double norm=0.0,lnorm, X[N][N];
    MPI_Datatype type;

    MPI_Comm_rank(MPI_COMM_WORLD, &rank);

    MPI_Type_vector(N, N, 3*N, MPI_DOUBLE, &type);
    MPI_Type_commit(&type);

    if (rank==0) {
       proc=1;
       for (i=0;i<3;i++)
         for (j=0;j<3;j++)
           if (i==0 && j==0)
              MPI_Sendrecv(&A[0][0], 1, type, 0, 100, X, N*N,
                    MPI_DOUBLE, 0, 100, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
           else {
              MPI_Send(&A[i*N][j*N], 1, type, proc, 100, MPI_COMM_WORLD);
              proc++;
           }
    } else
       MPI_Recv(X, N*N, MPI_DOUBLE, 0, 100, MPI_COMM_WORLD, MPI_STATUS_IGNORE);

    lnorm=0.0;
    for (i=0;i<N;i++)
      for (j=0;j<N;j++)
         lnorm += X[i][j]*X[i][j];

    MPI_Reduce(&lnorm, &norm, 1, MPI_DOUBLE, MPI_SUM, 0, MPI_COMM_WORLD);
```

```
        norm=sqrt(norm);
        MPI_Type_free(&type);
        return norm;
    }
```

0.3 p.

(b) Obtín el temps seqüencial i el temps paral·lel de la versió implementada, suposant que l'arrel quadrada té un cost de 5 flops.

Solució:

$$t(N) = \sum_{i=0}^{3N}\sum_{i=0}^{3N} 2 + 5 \approx 18N^2 flops$$

$$t(N,p) = t_a(N,p) + t_c(N,p)$$

$$t_a(N,p) = \sum_{i=0}^{N}\sum_{i=0}^{N} 2 + 5 \approx 2N^2 flops$$

$$t_c(N,p) = 8(t_s + N^2 t_w) + 8(t_s + t_w) + 8,$$

on el cost de $8(t_s + t_w) + 8$ correspon a l'operació de reducció (8 missatges d'un element i 8 flops per a les sumes). Per tant:

$$t(N,p) = 2N^2 + 8 + 16t_s + (N^2 + 8)t_w \approx 2N^2 + 16t_s + N^2 t_w$$