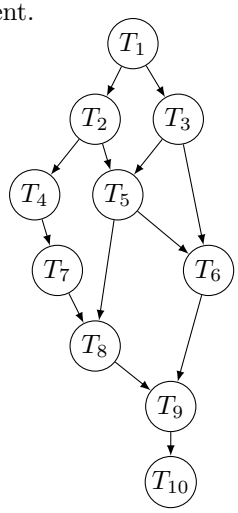
Qüestió 1 (1.2 punts)

Es vol paralelizar el següent fragment de codi mitjançant MPI amb 2 processos. Assumim que `n` és una constant predefinida. En totes les crides, el segon argument (el que va a continuació de `n`) és d'entrada-eixida, la qual cosa indueix les dependències entre tasques. Es proporciona el graf de dependències associat. El cost computacional de les funcions `f1`, `f2` i `f3` és de $\frac{1}{3}n^3$ flops, n^3 flops i $2n^3$ flops, respectivament.

```
double A[n][n], B[n][n], C[n][n], D[n][n],
       E[n][n], F[n][n];

f1(n,A);      /* Tasca T1 */
f2(n,D,A);    /* Tasca T2 */
f2(n,F,A);    /* Tasca T3 */
f2(n,B,D);    /* Tasca T4 */
f3(n,E,F,D);  /* Tasca T5 */
f3(n,C,E,F);  /* Tasca T6 */
f1(n,B);      /* Tasca T7 */
f2(n,E,B);    /* Tasca T8 */
f3(n,C,E,E);  /* Tasca T9 */
f1(n,C);      /* Tasca T10 */
```



0.9 p.

(a) Implementa la versió paral·lela amb MPI, utilitzant únicament primitives de comunicació punt a punt. Es pot suposar que inicialment totes les matrius contenen valors vàlids iguals en tots els processos. No és necessari que les matrius calculades s'arrepleguen en cap dels processos. L'assignació triada ha de fer, en primer lloc, que el càlcul estiga repartit raonablement entre els dos processos i, en segon lloc, que es minimitzen les comunicacions entre ells.

Solució: Per a repartir el càlcul entre els dos processos, assignarem les tasques T_2 i T_3 a processos diferents. També assignarem T_4 i T_7 a un procés i T_5 a un altre (atès que el cost de T_4 i T_7 juntes és menor que el de T_5). De la mateixa forma, assignarem T_6 i T_8 a processos diferents.

Per a minimitzar les comunicacions, haurien d'assignar-se les tasques T_2, T_4, T_7 i T_8 al mateix procés, mentre l'altre faria les tasques T_3, T_5 i T_6. A més, es replica la tasca T_1 per a evitar l'enviament d'un missatge.

D'acord amb això, l'assignació podria ser:

P_0: $T_1, T_2, T_4, T_7, T_8, T_9, T_{10}$.

P_1: T_1, T_3, T_5, T_6.

```
int rank;
MPI_Comm_rank(MPI_COMM_WORLD, &rank);

if (rank==0) {
  f1(n,A);     /* Tasca T1 */
  f2(n,D,A);   /* Tasca T2 */
  MPI_Send(D, n*n, MPI_DOUBLE, 1, 0, MPI_COMM_WORLD);
  f2(n,B,D);   /* Tasca T4 */
  f1(n,B);     /* Tasca T7 */
  MPI_Recv(E, n*n, MPI_DOUBLE, 1, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
  f2(n,E,B);   /* Tasca T8 */
  MPI_Recv(C, n*n, MPI_DOUBLE, 1, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
  f3(n,C,E,E); /* Tasca T9 */
  f1(n,C);     /* Tasca T10 */
} else if (rank==1) {
  f1(n,A);     /* Tasca T1 */
  f2(n,F,A);   /* Tasca T3 */
```

```
    MPI_Recv(D, n*n, MPI_DOUBLE, 0, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
    f3(n,E,F,D); /* Tasca T5 */
    MPI_Send(E, n*n, MPI_DOUBLE, 0, 0, MPI_COMM_WORLD);
    f3(n,C,F,F); /* Tasca T6 */
    MPI_Send(C, n*n, MPI_DOUBLE, 0, 0, MPI_COMM_WORLD);
  }
```

0.3 p. (b) Calcula el cost seqüencial i el cost paral·lel.

Solució: Cost seqüencial

$$t(n) = 3 \cdot \frac{1}{3}n^3 + 4 \cdot n^3 + 3 \cdot 2n^3 = 11n^3 \text{ flops}$$

Cost paral·lel

$$t_a(n,2) = \tfrac{1}{3}n^3 + n^3 + 2n^3 + 2n^3 + 2n^3 + \tfrac{1}{3}n^3 = (7 + \tfrac{2}{3})n^3 \text{ flops}$$
$$t_c(n,2) = 3(t_s + n^2 t_w)$$
$$t(n,2) = t_a(n,2) + t_c(n,2) = (7 + \tfrac{2}{3})n^3 \text{ flops} + 3(t_s + n^2 t_w)$$

Qüestió 2 (1.2 punts)

La següent funció implementa la següent expressió $y = A * (A * x)$, on A és una matriu quadrada de dimensió N i x un vector columna de la mateixa dimensió.

```
void fun1(double A[N][N], double x[], double y[]) {
    int i,j;
    double z[N];

    for (i=0;i<N;i++) {
       z[i]=0.0;
       for (j=0;j<N;j++)
          z[i]+=A[i][j]*x[j];
    }
    for (i=0;i<N;i++) {
       y[i]=0.0;
       for (j=0;j<N;j++)
          y[i]+=A[i][j]*z[j];
    }
}
```

0.8 p. (a) Implementa una versió paral·lela mitjançant MPI, assumint que les dades d'entrada es troben en el procés 0 i que els resultats han de trobar-se complets en aquest procés al final de l'execució. Es pot assumir que la grandària del problema és un múltiple del nombre de processos.

Solució:

```
void fun1_par(double A[N][N], double x[], double y[]) {
    int p, np;
    int i,j;
    double zlcl[N];
    double Alcl[N][N];

    MPI_Comm_size(MPI_COMM_WORLD, &p);

    np = N/p;
    MPI_Scatter(A, np*N, MPI_DOUBLE, Alcl, np*N, MPI_DOUBLE, 0, MPI_COMM_WORLD);
    MPI_Bcast(x, N, MPI_DOUBLE, 0, MPI_COMM_WORLD);

    for (i=0;i<np;i++) {
       zlcl[i]=0.0;
       for (j=0;j<N;j++)
          zlcl[i]+=Alcl[i][j]*x[j];
    }
    MPI_Allgather(zlcl, np, MPI_DOUBLE, y, np, MPI_DOUBLE, MPI_COMM_WORLD);
    for (i=0;i<np;i++) {
       zlcl[i]=0.0;
```

```
            for (j=0;j<N;j++)
               zlcl[i]+=Alcl[i][j]*y[j];
         }
         MPI_Gather(zlcl, np, MPI_DOUBLE, y, np, MPI_DOUBLE, 0, MPI_COMM_WORLD);
      }
```

0.4 p.

(b) Calcula l'expressió del temps paral·lel, així com el Speed Up i l'Eficiència. Calcula també els valors de Speed Up i eficiència quan la grandària del problema (N) tendeix a infinit. Indica per separat el cost de cada operació de comunicació realitzada, així com el temps aritmètic.

Solució: S'assumeix una implementació de Allgather en la qual es realitza un Gather en un procés i després un Broadcast de longitud N.

$$t(N) = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} 2 + \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} 2 = 2N^2 + 2N^2 = 4N^2$$

$$t(N,p) = t_{Scatter} + t_{Bcast} + t_{a1}(N,p) + t_{Allgather} + t_{Gather} + t_{a2}(N,p)$$

$$t_{Scatter} = (p-1)(t_s + \frac{N}{p}Nt_w)$$

$$t_{Bcast} = (p-1)(t_s + Nt_w)$$

$$t_{Allgather} = (p-1)(t_s + \frac{N}{p}t_w) + (p-1)(t_s + Nt_w)$$

$$t_{Gather} = (p-1)(t_s + \frac{N}{p}t_w)$$

$$t_a(N,p) = 2\sum_{i=0}^{\frac{N}{p}-1}\sum_{j=0}^{N-1} 2 = \frac{4N^2}{p}$$

$$t(N,p) = (p-1)(t_s + \frac{N}{p}Nt_w) + (p-1)(t_s + Nt_w) + \sum_{i=0}^{\frac{N}{p}-1}\sum_{j=0}^{N-1} 2 + (p-1)(t_s + \frac{N}{p}t_w) +$$

$$+(p-1)(t_s + Nt_w) + \sum_{i=0}^{\frac{N}{p}-1}\sum_{j=0}^{N-1} 2 + (p-1)(t_s + \frac{N}{p}t_w)$$

$$t(N,p) \approx 5pt_s + (N^2 + 2pN + 2N)t_w + 4\frac{N^2}{p} = 5pt_s + (N^2 + (2p+2)N)t_w + 4\frac{N^2}{p}$$

$$S(N,p) = \frac{t(N)}{t(N,p)} = \frac{4N^2}{5pt_s + (N^2 + (2p+2)N)t_w + 4\frac{N^2}{p}}$$

$$\lim_{N\to\infty} S(N,p) = \lim_{N\to\infty} \frac{t(n)}{t(n,p)} = \frac{4}{t_w + \frac{4}{p}}$$

$$E(N,p) = \frac{S(N,p)}{p} = \frac{4N^2}{p(5pt_s + (N^2 + (2p+2)N)t_w + 4\frac{N^2}{p})}$$

$$\lim_{N\to\infty} E(N,p) = \frac{4}{pt_w + 4}$$

Qüestió 3 (1.1 punts)

Siga una matriu $A \in R^{n^2\times n^2}$ amb blocs $A_i \in R^{n\times n}$, $i = 0, 1, \ldots, n-1$, situats al llarg de la diagonal principal de la matriu A, tal com es mostra en la següent figura, sent la resta dels elements iguals a 0:

$$A = \begin{pmatrix} A_0 & 0_{n\times n} & \cdots & 0_{n\times n} \\ 0_{n\times n} & A_1 & \cdots & 0_{n\times n} \\ & & \ddots & \vdots \\ 0_{n\times n} & 0_{n\times n} & \cdots & A_{n-1} \end{pmatrix}.$$

0.9 p.

(a) Implementa mitjançant MPI la funció `copia_blocs`, usant tipus derivats per a reduir el nombre de missatges enviats. Se suposa que la matriu A es troba emmagatzemada en el procés P_0 i ha de ser repartida

entre tots els processos, de manera que la matriu local B del procés P_i ha de contenir a la matriu A_i, $i = 0, 1, \ldots, n-1$. Suposeu que el nombre de processos és major o igual a n.

```
void copia_blocs(double A[n*n][n*n], double B[n][n]){
        ..........
}
```

Exemple d'una matriu A de dimensió 9×9 amb 3 processos:

$$A = \begin{pmatrix} 0 & 1 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3 & 4 & 5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 6 & 7 & 8 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 9 & 10 & 11 & 0 & 0 & 0 \\ 0 & 0 & 0 & 12 & 13 & 14 & 0 & 0 & 0 \\ 0 & 0 & 0 & 15 & 16 & 17 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 18 & 19 & 20 \\ 0 & 0 & 0 & 0 & 0 & 0 & 21 & 22 & 23 \\ 0 & 0 & 0 & 0 & 0 & 0 & 24 & 25 & 26 \end{pmatrix}$$

$$B(P_0) = \begin{pmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \\ 6 & 7 & 8 \end{pmatrix}, B(P_1) = \begin{pmatrix} 9 & 10 & 11 \\ 12 & 13 & 14 \\ 15 & 16 & 17 \end{pmatrix}, B(P_2) = \begin{pmatrix} 18 & 19 & 20 \\ 21 & 22 & 23 \\ 24 & 25 & 26 \end{pmatrix}.$$

Solució:

```
void copia_blocs(double A[n*n][n*n], double B[n][n]){
  int i, rank;
  MPI_Status stat;
  MPI_Datatype ntype;
  MPI_Comm_rank(MPI_COMM_WORLD, &rank);
  MPI_Type_vector(n,n,n*n,MPI_DOUBLE,&ntype);
  MPI_Type_commit(&ntype);
  if (rank==0){
    for(i=1; i<n; i++)
      MPI_Send(&A[i*n][i*n], 1, ntype, i, 100, MPI_COMM_WORLD);
    MPI_Sendrecv(&A[0][0], 1, ntype, 0, 100, B, n*n, MPI_DOUBLE,
                 0, 100, MPI_COMM_WORLD, &stat);
  }
  else if(rank<n)
    MPI_Recv(B, n*n, MPI_DOUBLE, 0, 100, MPI_COMM_WORLD, &stat);
  MPI_Type_free (&ntype);
}
```

0.1 p. (b) Calcula el temps de comunicacions de la funció `copia_blocs`.

Solució: Com el procés P_0 ha d'enviar el bloc A_i al procés P_i, $i = 1, \ldots, n-1$, el nombre total de missatges serà $n-1$, i com en cada missatge s'envien $n \times n$ dades, el temps de comunicacions és

$$t_c = (n-1)(t_s + n^2 t_w).$$

0.1 p. (c) Calcula el temps de comunicacions, suposant ara que no s'han usat tipus derivats en les comunicacions punt a punt.

Solució: En aquest cas, el procés P_0 ha d'enviar cadascuna de les n files de A_i en missatges separats al procés P_i, $i = 1, \ldots, n-1$, per tant el nombre total de missatges serà $(n-1)n$, i com en cada missatge s'envien n dades, el temps de comunicacions és

$$t_c = (n-1)n(t_s + n t_w).$$