Programación Paralela

Ecuación hiperbólica de onda

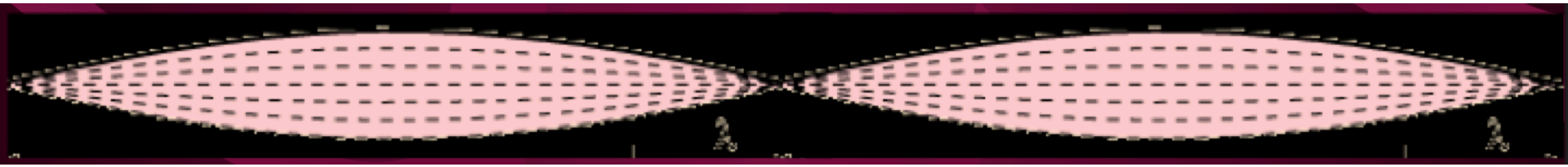
Prof. J.Fiestas

Ecuacion hiperbólica de onda:

La ecuación de onda es una ecuación diferencial de segundo orden, en una dimensión, que describe el movimiento de vibración de una cadena de tipo transversal o longitudinal.

$$\frac{\partial^2 u}{\partial t^2} = \frac{\partial^2 u}{\partial x^2}$$

u(x,t) representa la desviación al tiempo t de un punto de la cadena en la posición x en reposo

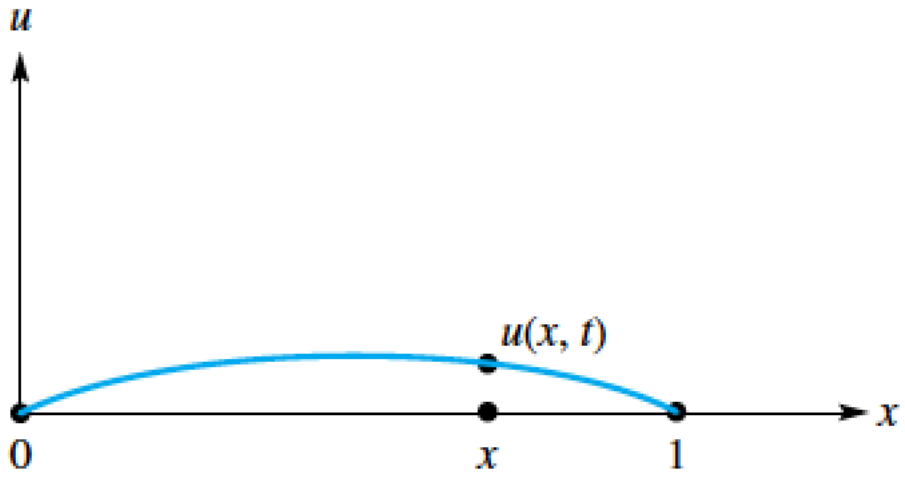


Ecuacion hiperbólica de onda:

Supongamos los puntos en la cadena tienen coordenadas en x entre 0<=x<=1, y que al tiempo t=0, la desviación satisface la ecuación u(x,0)=f(x)
Asimismo, los extremos de la cadena son fijos.

Es decir:

$$\begin{cases} u_{tt} - u_{xx} = 0 \\ u(x,0) = f(x) \\ u_t(x,0) = 0 \\ u(0,t) = u(1,t) = 0 \end{cases}$$



Ecuacion hiperbólica de onda:

Para encontrar una **solución analítica**, podemos postular la solución

$$u(x,t) = \frac{1}{2}[f(x+t) + f(x-t)]$$

siempre que f tenga dos derivadas, y se cumpla

$$f(-x) = -f(x) \qquad f(x+2) = f(x)$$

$$u(x,0) = f(x) \qquad u_t(x,0) = \frac{1}{2}[f'(x) - f'(x)] = 0$$

$$u(0,t) = \frac{1}{2}[f(t) + f(-t)] = 0$$

$$u(1,t) = \frac{1}{2}[f(1+t) - f(t-1+2)] = 0$$

Ecuacion hiperbólica de onda:

Para encontrar una **solución numérica**, con intervalos h para x e intervalos k para t

$$\frac{1}{h^2}[u(x+h,t)-2u(x,t)+u(x-h,t)]$$

$$=\frac{1}{k^2}[u(x,t+k)-2u(x,t)+u(x,t-k)]$$

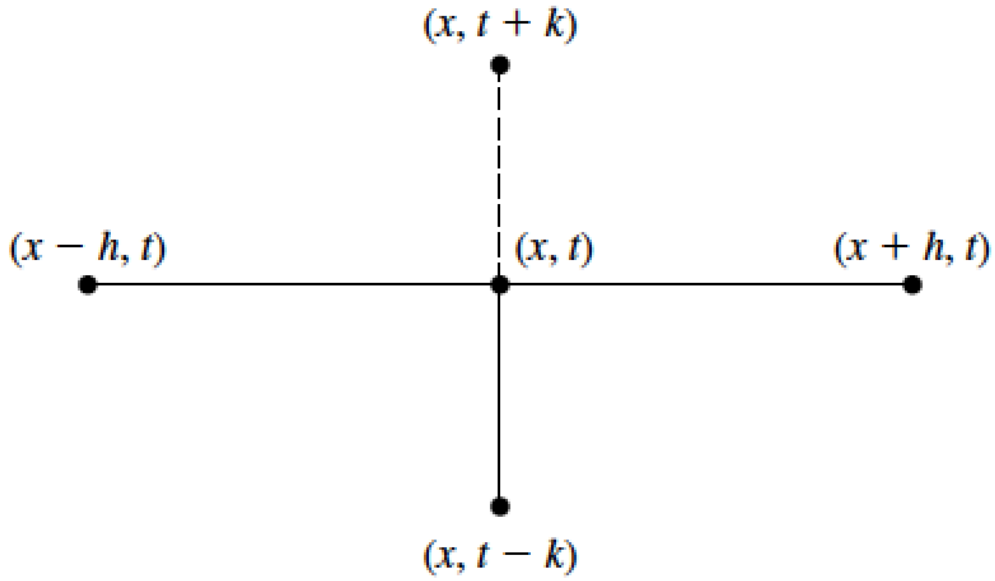
o simplificando:

$$u(x,t+k)=\rho u(x+h,t)+2(1-\rho)u(x,t)+\rho u(x-h,t)-u(x,t-k)$$

con $\rho=\frac{k^2}{h^2}$

Ecuacion hiperbólica de onda:

Para encontrar una solución numérica, con intervalos h para x e intervalos k para t



Con condiciones de frontera

$$\begin{cases} u(x,0) = f(x) \\ \frac{1}{k}[u(x,k) - u(x,0)] = 0 \\ u(0,t) = u(1,t) = 0 \end{cases}$$

Ecuacion hiperbólica de onda:

El problema será resuelto inicialmente en t=0, donde u(x,0)=f(x), y consecutivamente en t=k, t=2k, t=3k, ...

Note que $u(x,k) = u(x,0) = f(x)$

Para t=0, será

$$u(x,k) = \rho u(x+h,0) + 2(1-\rho)u(x,0) + \rho u(x-h,0) - u(x,-k)$$

Y finalmente: $$u(x,k) = \frac{1}{2}\rho[f(x+h) + f(x-h)] + (1-\rho)f(x)$$

Lo que permite calcular $u(x,nk), n \geqq 2.$ usando:

$$u(x,t+k) = \rho u(x+h,t) + 2(1-\rho)u(x,t) + \rho u(x-h,t) - u(x,t-k)$$

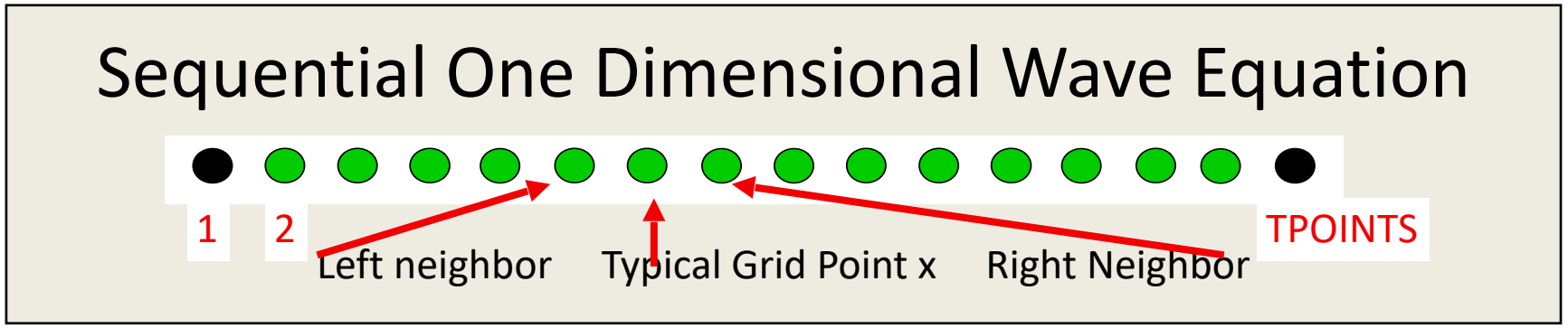
Ecuacion hiperbólica de onda:

Discretizando:

$$(u(x,t+1)-2u(x,t)+u(x,t-1))/\delta t^2 = -c^2\,(u(x+1,t)-2u(x,t)+u(x-1,t))/\delta x^2$$

O tambien, para calcular u en un paso en el futuro, basado en x actual y un paso en el pasado

$$u(x,t+1) = 2u(x,t) - u(x,t-1) - (u(x+1,t)-2u(x,t)+u(x-1,t))\,(c^2\,\delta t^2/\delta x^2)$$



Programa en serie para resolver la ecuacion de onda:

```
#define MAXPOINTS 1000
#define MAXSTEPS 1000
#define MINPOINTS 20
#define PI 3.14159265
int nsteps,                 /* number of time steps */
        tpoints;            /* total points along string */
double values[MAXPOINTS+2],     /* values at time t */
        oldval[MAXPOINTS+2],    /* values at time (t-dt)*
        newval[MAXPOINTS+2];    /* values at time (t+dt) */
int main(int argc, char *argv[])
{
init_param();
init_line();
update();
printfinal();
}
```

Definición de parametros

```
void init_param(void)
  {
  char tchar[8];
  /* set number of points, number of iterations */
  tpoints = 0;
  nsteps = 0;
  while ((tpoints < MINPOINTS) || (tpoints > MAXPOINTS))  {
    printf("Enter number of points along vibrating string [%d-%d]: ”, MINPOINTS,
MAXPOINTS);
    scanf("%s", tchar);
     tpoints = atoi(tchar);
    }

  while ((nsteps < 1) || (nsteps > MAXSTEPS)) {
    printf("Enter number of time steps [1-%d]: ", MAXSTEPS);
   scanf("%s", tchar);
     nsteps = atoi(tchar);
    }

  printf("Using points = %d, steps = %d\n", tpoints, nsteps);
  }
```

Inicialización de puntos en línea, de acuerdo a función sin()

```
void init_line(void) {
  int i, j;
  double x, fac, k, tmp;
  /* Calculate initial values based on sine curve */
  fac = 2.0 * PI;
  k = 0.0;
  tmp = tpoints - 1;
  for (j = 1; j <= tpoints; j++) {
    x = k/tmp;
    values[j] = sin (fac * x);
    k = k + 1.0;
    }
  /* Initialize old values array */
  for (i = 1; i <= tpoints; i++)
    oldval[i] = values[i];
  }
```

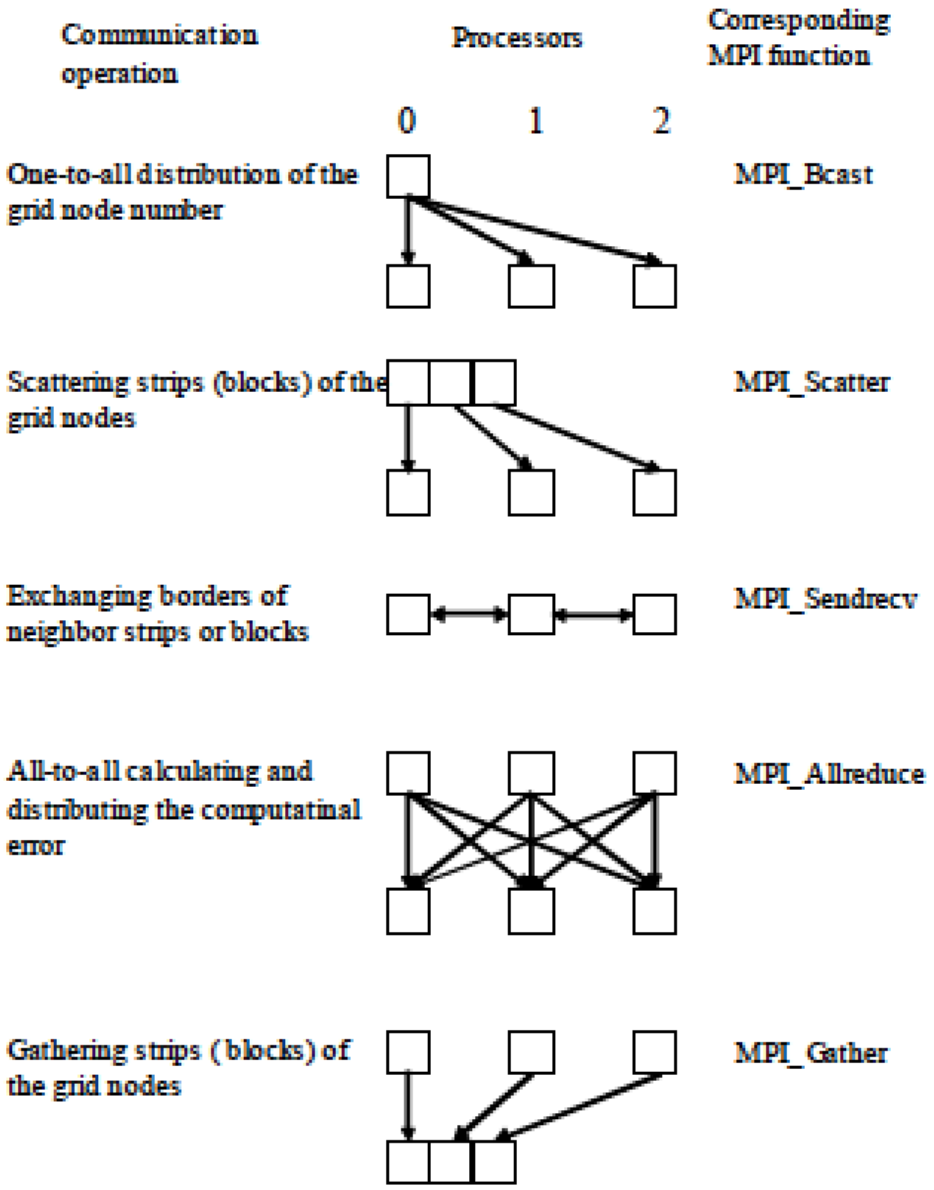
Actualizar los valores en una linea un número determinado de veces

```
void update()
  {
  int i, j;
  /* Update values for each time step */
  for (i = 1; i<= nsteps; i++) {
    /* Update points along line for this time step */
    for (j = 1; j <= tpoints; j++) {
      /* global endpoints */
      if ((j == 1) || (j  == tpoints))
        newval[j] = 0.0;
      else
        do_math(j);
      }
    /* Update old values with new values */
    for (j = 1; j <= tpoints; j++) {
      oldval[j] = values[j];
      values[j] = newval[j];
      }
    }
  }
```

Calculo de nuevos valores a partir de la ecuacion de onda

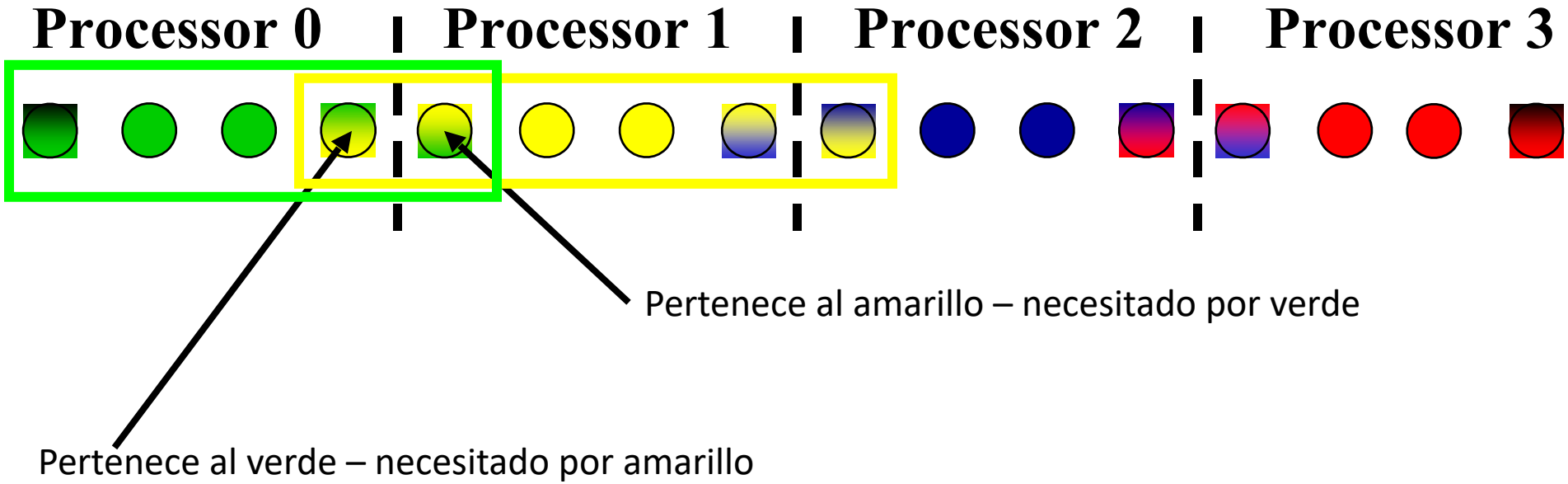
```
void do_math(int i)
  {
  double dtime, c, dx, tau, sqtau;

  dtime = 0.3;
  c = 1.0;
  dx = 1.0;
  tau = (c * dtime / dx);
  sqtau = tau * tau;
  newval[i] = (2.0 * values[i]) -oldval[i]
+ (sqtau * (values[i-1] - (2.0 * values[i])
+ values[i+1]));

  }
```

Comunicación de data para la solución de ecuaciones diferenciales



Ecuacion hiperbólica de onda:

Esquema de algoritmo en paralelo



Necesita comunicación P2P para intercambiar puntos de frontera con vecinos

```
void update(int left, int right) {
.....
  /* Update values for each point along string */
  for (i = 1; i <= nsteps; i++) {
    /* Exchange data with "left-hand" neighbor */
    if (first != 1) {
      MPI_Send(&values[1], 1, MPI_DOUBLE, left, RtoL, MPI_COMM_WORLD);
      MPI_Recv(&values[0], 1, MPI_DOUBLE, left, LtoR, MPI_COMM_WORLD,
            &status);
      }
    /* Exchange data with "right-hand" neighbor */
    if (first + npoints -1 != TPOINTS) {
      MPI_Send(&values[npoints], 1, MPI_DOUBLE, right, LtoR, MPI_COMM_WORLD);
      MPI_Recv(&values[npoints+1], 1, MPI_DOUBLE, right, RtoL,
            MPI_COMM_WORLD, &status);
      }
 /* Update points along line */
   .....
    }
  }
```

Ejercicio:

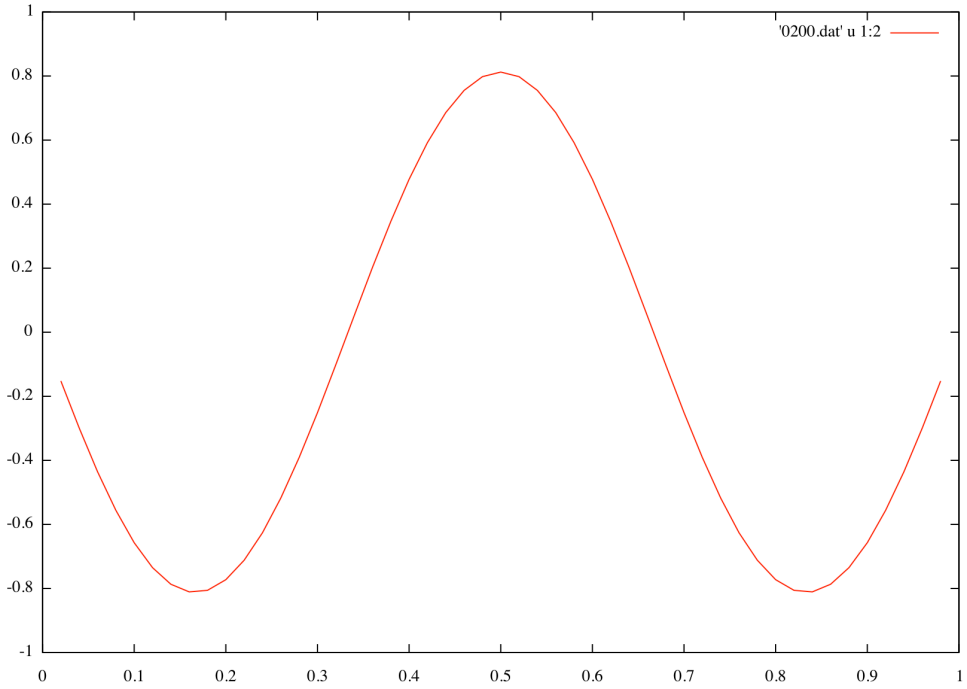
1. **Compilar y ejecutar el código secuencial de vibración de onda en una dimensión (onda_sec.c)**

Utilizar tpoints=800 puntos, nsteps=1000.

La función inicial es

$$f(x)=\sin(2\pi x).$$

Condiciones de frontera son values(0)= 0 values(tpoints)=0



Ejercicio:

2. Paralelizar el código utilizando el algoritmo discutido en clase

Para ello:

- Repartir el dominio en partes iguales entre los procesos
- Utlizar comunicación P2P (MPI_Send, MPI_Recv) para intercambiar valores de frontera con vecinos, y actualizar los puntos de cada proceso
- Enviar valores de procesos al nodo maestro, guardarlos en un vector, imprimirlos y graficar la onda de una dimension

Ejercicio:

- Medir tiempos de ejecución para np=2,4,8 utilizando los mismos parámetros de simulación.

3. Opcional

Animar la vibración de la cadena en el tiempo para un número determinado de nsteps.