High Performance Computing
Departamento de Ingeniería en Informática
LAB2 : SIMD- OpenMP

1 Objetivo

El objetivo de este laboratorio es implementar un simulador paralelo de la difusión de una onda según la equación de Schroendinger, usando OpenMP como tecnología de paralelización.

2 La ola

2.1 La ecuación de Schroedinger

Suponga que deseamos simular cómo se difunde una ola en un medio, por ejemplo, acuoso, a partir de un estado inicial. Para simplificar, discretizamos el dominio de difusión en una grilla de $N \times N$ celdas. En particular, la matriz H contiene la altura de la onda en cada posición de la grilla. Además, como el proceso de difusión es dependiente del tiempo, H^t_{ij} es la altura, ya sea positiva o negativa, de la onda en la celda (i, j), en tiempo t. La ecuación de Schroendinger dice que:

$$H^t_{i,j} = 2H^{t-1}_{i,j} - H^{t-2}_{i,j} + c^2\Big(\frac{dt}{dd}\Big)^2\Big(H^{t-1}_{i+1,j} + H^{t-1}_{i-1,j} + H^{t-1}_{i,j-1} + H^{t-1}_{i,j+1} - 4H^{t-1}_{i,j}\Big) \tag{1}$$

donde c es la velocidad de la onda en el medio, dt es el intervalo de tiempo con que avanza la simulación y dd es el cambio en la superficie. Para este lab, usaremos $c = 1.0$, $dt = 0.1$, y $dd = 2.0$.

2.2 Condiciones de borde y especiales

La **condición de borde** para este problema es cero, es decir la altura en las celdas de los bordes del dominio permanece siempre con valor cero,

$$H^t_{0,j} = H^t_{i,0} = H^t_{N-1,j} = H^t_{i,N-1} = 0 \quad \forall i, j, t$$

También, la iteración para $t = 1$, es la siguiente ecuación:

$$H^t_{i,j} = H^{t-1}_{i,j} + \frac{c^2}{2}\Big(\frac{dt}{dd}\Big)^2\Big(H^{t-1}_{i+1,j} + H^{t-1}_{i-1,j} + H^{t-1}_{i,j-1} + H^{t-1}_{i,j+1} - 4H^{t-1}_{i,j}\Big) \tag{2}$$

En la ecuación anterior se respeta la condición de borde del problema.

La **condición inicial** del sistema está dada por un impulso en celdas centrales de la grilla. Por ejemplo, para una ola cuadrada inicial, tenemos:

$$H^0_{i,j} = 20 \quad \forall\ 0.4N < i < 0.6N,\ \ 0.4N < j < 0.6N$$

y $H^0_{i,j} = 0$ para todas las otras celdas.

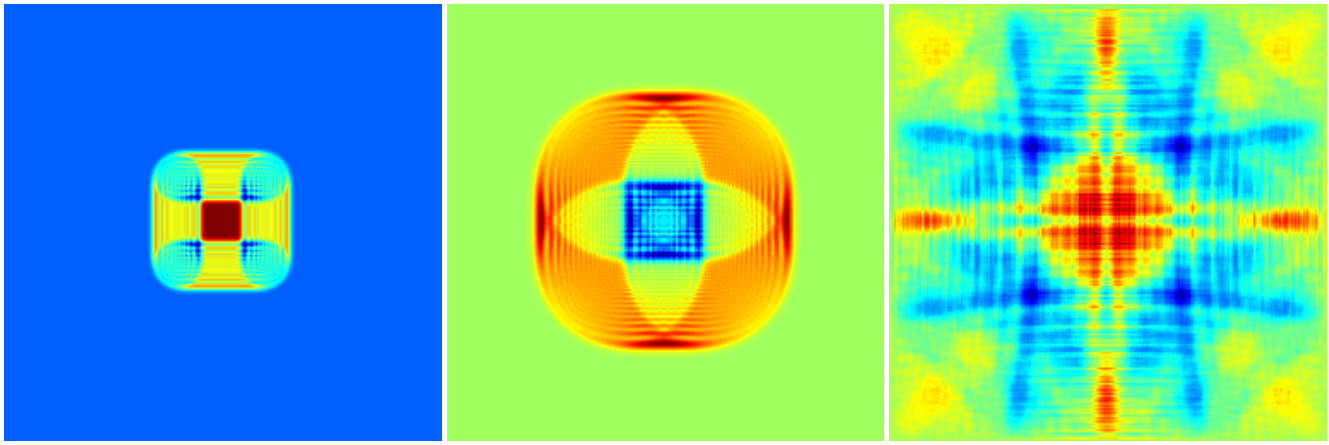

Figure 1: Imágenes de Schroedinger para $t = 300, 1000$, y 10000, para una grilla de 256×2565.

3 El programa y la estrategia de paralelización

OpenMP es la herramienta perfecta para este lab. Elija una estrategia de paralelización apropiada a este tipo de problemas.

El programa se ejecutará de la siguiente manera

```
$ ./wave -N tamaño_grilla -T número_de_pasos -H número_de_hebras
          -f archivo_de_salida -t iteracion_de_salida
```

Note que el número de hebras NO necesariamente divide exactamente al tamaño de la grilla. En el archivo de salida se almacenará la grilla de la iteración indicada por la opción `t`, en formato binario, es decir H^t.

El **tipo de dato para la grila** es `float`.

Las imágenes de la Figura 1 muestran ejemplos de salida para una grilla de 256×256, en diversa iteraciones.

4 Matlab

Se recomienda programar este problema, primero en Matlab, y luego en C con OpenMP. Esto le ayudará a tener en poco tiempo un prototipo secuencial funcionando, y además usar la facilidades de graficación de Matlab.

El archivo de salida de su programa estará en formato binario (.raw), es decir los valores son almacenados en disco tal cual se representan en el procesador. Por ejemplo, si usa flotantes de precisión simple (`float`), una posible forma de procurar memoria para una matriz sería:

```
float *H = malloc(N*N*sizeof(float));
```

Note que `H[]` es simplemente una secuencia de flotantes y para acceder a la posición (i, j), se debe usar

```
H[i*N + j] = ...
```

Luego, para guardar dicha *imagen* en un archivo, basta con:

```
FILE *f = fopen("ejemplo.raw", "w");
fwrite(H, sizeof(float), N*N, f);
fclose(f);
```

Para visualizar esta imagen podemos usar matlab, de la siguiente forma:

```
>> N = 256; % depende del tamaño de la imagen
>> f = fopen('ejemplo.raw', 'r');
>> I = fread(f, N*N, 'float32');
>> I = reshape(I, N, N);
>> I = I'; % simplemente traspone la matriz
```

Finalmente, para visualizar la imagen:

```
>> imagesc(H); axis('off');axis('square');
```

Si desea escribir la matriz en formato binario y en floats, entonces

```
>> f = fopen('miarchivo.raw', 'w');
>> fwrite(f, H', 'float');
>> fclose(f);
```

Note que la función `fwrite()` recibe como parámetro la traspuesta de la matriz `A`, pues Matlab al igual que Fortran almacena las matrices por columnas, no por filas.

5 Entregables

Tarree, comprima y envíe a `fernando.rannou@usach.cl` al menos los siguientes archivos:

1. `Makefile`: archivo para make que compila los programas
2. `wave.c`: archivo con el código. Puede incluir otros archivos fuentes.
3. `wave.pdf`: informe breve de su solución. Estrategia de paralelizaciín y rendimiento computacional.

Fecha de entrega: domingo 09 de octubre antes de las 23:59 hrs.