Universidad Nacional Autónoma de México

Posgrado en Ciencias e Ingeniería de la Computación

PARALELIZACIÓN DEL ALGORITMO SEMI-IMPLICITO PARA EL MODELADO DEL MOVIMIENTO DE PARTÍCULAS

PROYECTO FINAL

PARA OPTAR POR EL GRADO DE:

ESPECIALISTA EN COMPUTO DE ALTO RENDIMIENTO

P R E S E N T A :

ARTURO AVILA ROSAS

Director:

Dr. Luis Miguel de La Cruz Salas

Instituto de Geofísica UNAM

Codirector:

Dr. Joel Sánchez Mondragón

Instituto Mexicano del Petróleo

Ciudad Universitaria, CDMX 2019

Contenido

1 Introducción

Simular el comportamiento real de fluidos en movimiento bajo circunstancias de alta no linealidad es una tarea compleja. Por ejemplo, cuando se trata de simular el movimiento periódico de la superficie de un líquido que es transportado en un recipiente y cuya frecuencia en cierto momento coincide con el periodo de resonancia, algo similar ocurre al abordar los fenómenos de fragmentación y coalescencia en fluidos. Para el tratamiento numérico de esta clase de problemas es necesario aplicar métodos numéricos que no padezcan de las limitaciones de los métodos tradicionales de la Dinámica de Fluidos Computacional, como el Método del Volumen Finito (MVF). Recientemente Koshizuka y Oka (1996) y Koshizuka *et al.* (1998) desarrollaron el método de Movimiento de Partículas Semi-implícito (MPS) que se basa en el método Smoothed Particle Hydrodynamics (SPH) de Monaghan (1992) el cual es un método Lagrangiano libre de mallas. Estos métodos se pueden aplicar de forma simple a la simulación de la superficie libre de fluidos en fenómenos tan complejos como el rompimiento de oleaje, en donde el comportamiento del fluido suele ser violento. El método MPS es capaz de modelar fielmente el movimiento de oleaje y estimar la presión que ejercen las olas al chocar contra superficies solidas como estructuras marinas; este fenómeno es difícil de representar numéricamente si se emplean otro tipo de modelos. En el método MPS se calcula la posición y la velocidad de cada una de las partículas que representan a un fluido en movimiento, para lo cual se usan ecuaciones complejas que capturan la interacción entre las partículas y con ello es posible describir el movimiento general del fluido.

Hoy en día el uso del método MPS para modelar diferentes problemas de ingeniería va en ascenso, y se ha empleado para modelar el aceite en cajas de cambios, los fluidos en tanques mezcladores, los procesos de enfriamiento en soldaduras e inclusive tsunamis.

Por otro lado, durante los últimos años el desarrollo de computadoras de alta velocidad de procesamiento ha modificado la forma en que se aplican los principios de la mecánica de fluidos a la solución de los problemas de diseño en la práctica de la ingeniería moderna. El impacto que ha provocado la presencia de computadoras con un gran poder computacional, especialmente las de *procesamiento paralelo*, en la modelación de fluidos se manifiesta en una reducción importante del tiempo de procesamiento y en la resolución detallada de las simulaciones que son capaces de predecir de manera más precisa un fenómeno. Los problemas que tardaban varios años en resolverse usando los métodos y computadoras disponibles hace 50 años, hoy en día se resuelven a un costo muy bajo y en tan solo unos segundos. En cuanto a la resolución, el cálculo de algunos fenómenos físicos, como el flujo de fluidos o la predicción meteorológica, implica *discretizar* el problema para obtener una mejor aproximación de los fenómenos simulados, aunque esto implica aumentar la cantidad de cálculos a realizar. Mientras que en el pasado las soluciones eran burdas, en cierto grado limitadas por la capacidad de las computadoras tradicionales de un solo procesador, hoy en día las computadoras con *múltiples núcleos* son capaces de efectuar un gran número de cálculos en paralelo y obtener soluciones más precisas donde es posible distinguir detalles que antes no era posible advertir.

Ahora bien, dado que en el método MPS el cálculo de la velocidad y posición de n partículas se repite k veces para simular el movimiento de un fluido, no es difícil concluir que el método MPS es un ejemplo claro de aplicación que requiere un poder computacional intenso. Este trabajo está dedicado a paralelizar el algoritmo numérico del método MPS. Dado que existen diferentes estrategias de parelización

asociadas con el tipo hardware existente, que van desde supercomputadoras hasta laptops con multiprocesadores, en este trabajo se optó por construir un programa en la arquitectura de *memoria compartida*, específicamente computadoras *múltiples núcleos*. Dos son las razones, la primera tiene ver con la relación precio-rendimiento de este tipo de arquitecturas, a un costo bajo se pueden obtener más de dos procesadores que tienen acceso a todos los datos sin que se necesiten comunicarse explícitamente. La segunda razón es por la amplia disponibilidad de este tipo de arquitecturas donde es posible realizar actividades de cómputo científico usando programación en paralelo en su forma más básica antes de migrar hacia arquitecturas más especializadas si así se requiere. Para concluir, en este trabajo se usó *OpenMP*, el cual hoy en día es el estándar dominante de programación para arquitecturas de memoria compartida, exceptuando, claro está, las arquitecturas basadas en GPUs donde CUDA y OpenCL dominan.

En el resto de este capítulo se describen los métodos Lagrangianos y Eulerianos, y sus diferencias en el modelado de fluidos. En el capítulo 2 se describe el método MPS incluyendo las ecuaciones que permiten discretizar la solución de las ecuaciones que gobiernan el movimiento de fluidos. El capítulo 3 ofrece una descripción del algoritmo del método MPS y de la programación serial del mismo. El capítulo 4 discute la paralelización del método MPS con OpenMP tomando como punto de partida la salida del perfilador gprof. Los resultados de la simulación usando diferentes *hilos* de paralelización se presentan en el capítulo 5. El capítulo 6 contiene las conclusiones que se obtuvieron del presente trabajo. Finalmente, el capítulo 6 incluye las referencias consultadas para la elaboración de este trabajo.

1.1 Métodos de mallas y de partículas

Las ecuaciones que gobiernan la mecánica del medio continuo son las de conservación de la masa y de conservación de momento. En dinámica de fluidos, se conocen como ecuaciones de continuidad y de Navier-Stokes respectivamente y su solución se puede aproximar mediante algún método numérico que emplea la forma discreta de las ecuaciones. La discretización puede ser espacial o material, lo que da lugar a los métodos de mallas (descripción Euleriana) y a los métodos libres de mallas (descripción Lagrangiana).

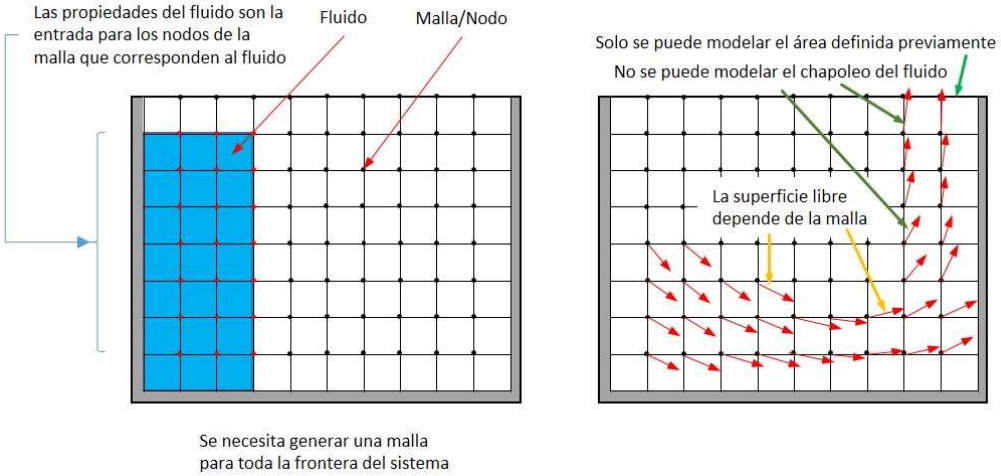


Figura 1.1. Conceptualización grafica de los métodos tradicionales basados en mallas, e.g., Método del elemento finito y Método del volumen finito.

Los métodos del volumen finito y del elemento finito son métodos de mallas clásicos que se usan en dinámica de fluidos computacional DFC (Computational Fluids Dynamics) y en mecánica de solidos computacional respectivamente. Los métodos de partículas, SPH (Smoothed Particle Hydrodynamics) y MPS se consideran una nueva alternativa a los métodos convencionales de mallas. El método SPH fue desarrollado en 1977 por Robert Gingold y Joseph Monaghan y, aunque en un principio se usó para resolver problemas de astrofísica, en 1994 Monaghan extendió su aplicación a la simulación de fluidos. En las figuras 1.1 y 1.2 se puede observar de forma gráfica el concepto de dicretización que emplean los métodos Eulerianos y Lagrangianos, respectivamente.

Los métodos de partículas pueden modelar más fácilmente las superficies libres de los fluidos, inclusive se pueden modelar varios tipos de fluidos juntos, así como la frontera entre esos fluidos. Otra ventaja consiste en modelar fluidos con fronteras geométricamente complejas. En contraste, los métodos convencionales DFC es difícil simular la superficie libre de un fluido o fronteras con geometría compleja.

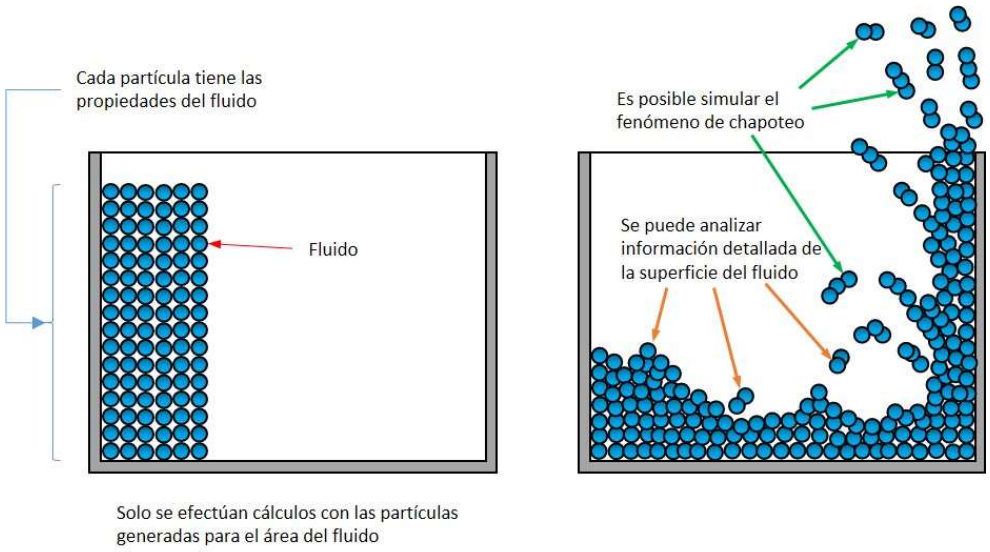


Figura 1.2. Conceptualización grafica de los métodos basados en partículas, e.g., MPS y SPH.

1.2 Descripción Lagrangiana

Las partículas son los puntos materiales de evaluación donde se localizan las variables de interés como la masa, las componentes del vector de velocidad, la presión, la temperatura, etc., de aquí que las partículas se consideren como el equivalente a los nodos de los métodos de mallas. La diferencia radica en que las partículas se mueven con sus velocidades, mientras que los nodos permanecen estáticos. En la descripción Lagrangiana, las partículas están atadas al movimiento de la materia y en la descripción Eulerania los nodos de la malla están fijos en el espacio, de esta manera en la descripción Lagrangiana las propiedades de la materia se mueven con la partícula, mientras que en la descripción Euleriana no.

Los métodos de partículas requieren, por lo tanto, tener control de las componentes del vector de posición y del vector de velocidad. Esto significa que, contrario a los métodos de malla, se emplea tiempo adicional de cálculo para actualizar la posición de cada partícula en cada paso de tiempo de una simulación.

Una partícula en movimiento debe ser vista como una sustancia de un material con masa constante, y cada división del material viaja con su propia velocidad. Tanto el vector de velocidad como el de posición se consideran aplicados en el centro de gravedad de cada división del material. Debido a que la distribución de masa no se considera en la partícula tampoco lo es la superposición de partículas.

Un punto importante en los métodos de partículas es que la conservación de masa se satisface esencialmente al mantener el número de partículas inicial. Entonces, si se agregan partículas significa que la masa aumenta y si se eliminan partículas la masa decrece. Esta característica es muy valiosa en el modelado de fluidos incompresibles. Por el contrario, en los métodos de malla se debe derivar la conservación de la masa para satisfacer la ecuación de continuidad la cual representa la relación entre la masa y el espacio.

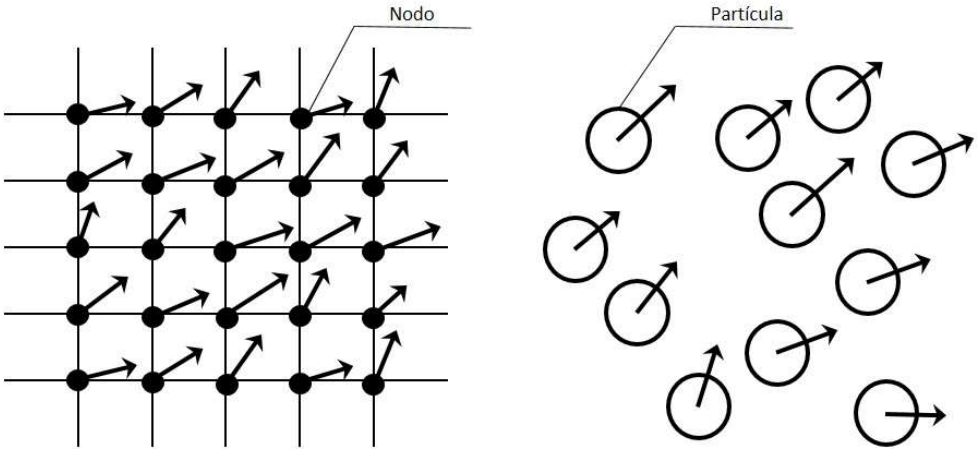


Figura 1.3. Método de mallas y método de partículas.

Las ecuaciones discretización surgen de la relación de nodos que son adyacentes. Por ejemplo, en el espacio bidimensional de la figura 1.3 para un nodo con cuatro segmentos que conectan a cuatro nodos adyacentes, se construye un sistema de ecuaciones a partir del cálculo de las diferencias finitas usando las variables localizadas en los 5 nodos. Es claro entonces que la malla ayuda a identificar los nodos adyacentes que se usan para discretizar las ecuaciones gobernantes, y la malla muestra explícitamente las adyacencias.

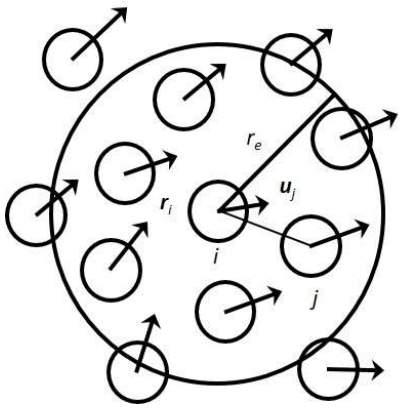


Figura 1.4. Interacción de una partícula con las partículas vecinas.

En los métodos de partículas no existen tales segmentos y por lo tanto no se muestran las partículas adyacentes explícitamente. Las partículas adyacentes se identifican solo durante la simulación al evaluar la distancia entre dos partículas, ver figura 1.4. Si la distancia r_{ij} $(=|\boldsymbol{r}_i - \boldsymbol{r}_j|)$ entre dos partículas *i* y *j*, con los vectores de posición $\boldsymbol{r}_i$ y $\boldsymbol{r}_j$ respectivamente, es menor que el radio efectivo $\boldsymbol{r}_e$ entonces se consideran adyacentes, y se dice que la partícula *j* adyacente es una partícula vecina. En cada paso de tiempo de la simulación se construye una lista de vecinos para cada partícula calculando las distancias de todas las combinaciones de pares de partículas. Las ecuaciones de discretización se construyen usando las variables en las partículas vecinas.

Es importante destacar que, las partículas son artificiales durante la discretización, un fluido se representa por un numero finto de partículas artificiales. No se trata de partículas reales como el polvo o moléculas. Un ejemplo de la dinámica de partículas reales es el movimiento de cuentas o lentejuelas, ver figura 1.5. Se puede observar del movimiento de las cuentas cuando se vierten en un vaso, que la dinámica de las cuentas se describe con la segunda ley de Newton agregando los términos debidos a la fricción, figura 1.5A.

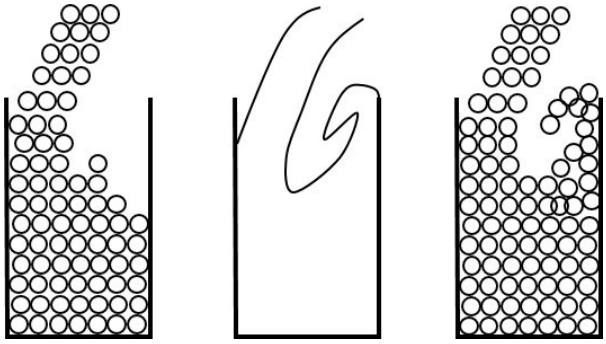


Figura 1.5. Dinámica de las partículas: A) movimiento real de cuentas, B) movimiento real del agua y C) movimiento simulado del agua usando el método de partículas.

Por otro lado, las ecuaciones que gobiernan el movimiento del agua cuando se vierte sobre un vaso son las ecuaciones de Navier-Stokes, figura 1.5B. El comportamiento del agua es diferente al de las cuentas porque las ecuaciones gobernantes son diferentes. Sin embargo, como se muestra en la figura 1.5C, el movimiento de las partículas se evalúa usando la discretización de las ecuaciones de Navier-Stokes y de esta manera se consigue un movimiento similar al del agua. Este es el concepto de los métodos de partículas usando partículas artificiales en Mecánica del medio continuo.

De lo anterior se puede decir que los métodos que emplean mallas tienen dificultades para representar correctamente el rompimiento de la superficie libre del fluido. Por otro lado, los métodos de partículas pueden analizar fácilmente superficies libres, esto se debe a que siguen el movimiento de cada partícula del fluido a lo largo de toda la simulación. Cambios en la presión y densidad de las partículas vecinas se pueden calcular a partir de la posición y velocidad de cada partícula del fluido en cada instante de tiempo.

2 Método de Movimiento de Partículas Semi-implicito MPS

El método fue propuesto por Koshizuka y Oka (1995) para analizar flujos incompresibles, como el agua y el aceite. Un fluido se clasifica como *compresible* o *incompresible*, dependiendo del nivel de variación de la densidad del flujo en ese flujo. La incompresibilidad es una aproximación y se dice que el flujo es incompresible si la densidad permanece aproximadamente constante a lo largo de todo el flujo. Por lo tanto, el volumen de todas las porciones del fluido permanece inalterado sobre el curso de su movimiento cuando el flujo se modela como incompresible.

En esencia, las densidades de los líquidos son constantes y, así, el flujo de ellos es típicamente incompresible. La mayoría de los fluidos analizados/estudiados en aplicaciones de ingeniería se considera incompresible, porque la velocidad del sonido es más alta que la velocidad de flujo.

2.1 Ecuaciones que gobiernan el movimiento de un fluido

Las ecuaciones que gobiernan el movimiento de un fluido viscoso son las ecuaciones de continuidad (conservación de la masa) y de Navier-Stokes (conservación del momentum). La ecuación de continuidad se puede escribir como:

$$\frac{D\rho}{Dt} + \nabla \cdot (\rho \vec{u}) = 0 \qquad (1)$$

para fluidos incompresibles se reduce a:

$$\frac{1}{\rho}\frac{D\rho}{Dt} = 0 \qquad (2)$$

y la de Navier-Stokes:

$$\rho \frac{D\vec{u}}{Dt} = -\nabla P + \mu \nabla^2 \vec{u} + \rho \vec{g} \qquad (3)$$

donde ρ y $\vec{u}$ son la densidad del fluido y el vector de velocidad respectivamente. $\frac{D}{Dt}$ es la derivada material y al aplicarse sobre la velocidad expresa la aceleración de la partícula. P, μ y $\vec{g}$ son la presión, el coeficiente de viscosidad y la aceleración de la gravedad respectivamente. ∇ es el operador nabla y ∇^2 es el Laplaciano. El lado derecho está compuesto por las fuerzas que actúan sobre el fluido.

2.2 Diferencias pesadas

La diferencia entre las cantidades ϕ_i y ϕ_j de dos partículas i y j se calcula como se ilustra en la figura 2.1, el resultado se muestra entre las dos partículas, donde ϕ_i denota la representación escalar de una cantidad física de la partícula i que está ubicada en la posición x_i. Este es el concepto básico de la discretización espacial en los métodos de partículas. Usualmente se considera un radio de interacción entre partículas finito, de tal forma que la interacción se supone solo entre las partículas más cercanas. Esta limitación ayuda a reducir el tiempo de cómputo basado en el argumento de que las fuerzas de interacción con las partículas alejadas se pueden despreciar sin que esto afecte el comportamiento del fluido.

La diferencia de primer orden en la partícula i en un espacio unidimensional se calcula como el promedio pesado de las diferencias con sus partículas vecinas:

$$\langle \frac{d\phi}{dx} \rangle_i = \frac{1}{n_i} \sum_{j \neq i} \frac{\phi_j - \phi_i}{x_j - x_i} w(|x_j - x_i|), \qquad (4)$$

donde $w(r)$ es una función de peso o kernel:

$$w(r) = \begin{cases} \frac{r_e}{r_{ij}} - 1 & r_{ij} \leq r_e \\ 0 & r_{ij} > r_e \end{cases}, \qquad (5)$$

y n_i es la sumatoria de los valores de la función de peso:

$$n_i = \sum_{j \neq i} w(|x_j - x_i|), \quad (6)$$

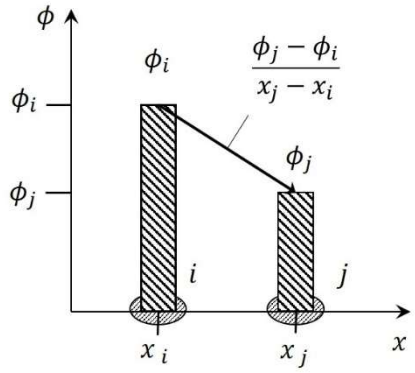


Figura 2.1. Diferencia entre dos partículas.

La ecuación (5) es una función de peso típica en el método MPS como se ilustra en la figura 2.2. La función de peso w decrece con r hasta llegar a $w = 0$ en $r_{ij} \geq r_e$, además es infinita en $r_{ij} = 0$. El valor infinito es preferible y previene la formación de cúmulos de partículas a través de la condición de incompresibilidad.

2.3 Densidad del número de partículas

En el método MPS, se usa la densidad del número de partículas n_i para evaluar la densidad de un fluido. La densidad del número de partículas refleja el número de partículas alrededor de una partícula y se define como la suma de los pesos de las partículas vecinas. La densidad del número de partículas de una partícula concreta expresa la densidad de partículas en la posición en que se encuentra. La densidad del número de partículas de la partícula i se calcula con la siguiente ecuación:

$$n_i = \sum_{j \neq i} w(|\vec{r}_j - \vec{r}_i|), \quad (7)$$

donde n_i es la densidad del número de partículas de la partícula *i*, $\vec{r}_i$ es el vector de posición de la partícula *i*, y $\vec{r}_j$ son los vectores de posición de las partículas *j* y $w(r)$ es la función de peso.

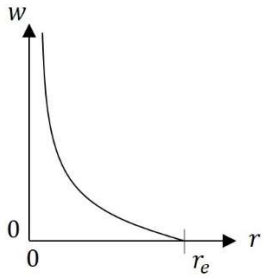


Figura 2.2. Función de peso o *kernel*.

La densidad del número de partículas estándar n^0es un valor constante y se calcula en el estado inicial ($t = 0s$) cuando las partículas están arregladas en intervalos regulares usando la siguiente formula:

$$n^0 = \sum_{j \neq i} w\left(\left|\vec{r}_j^{\,0} - \vec{r}_i^{\,0}\right|\right), \quad (8)$$

Se usa n^0 como una constante para todas las partículas durante la simulación y es también el valor estándar de la suma total de pesos para una partícula interior del fluido. Además, se usa para normalizar la función de peso de un promedio pesado.

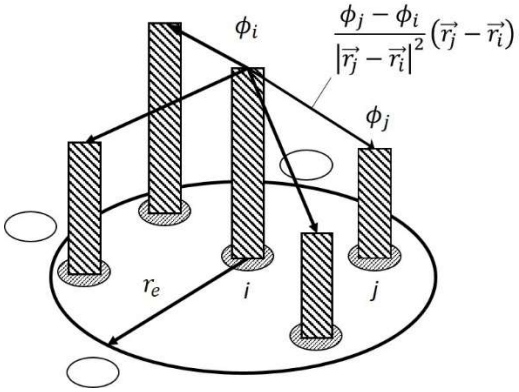


Figura 2.3. Modelo de gradiente.

2.4 Modelos de interacción de la partícula

En el método MPS, los modelos de interacción de la partícula se obtienen de los operadores diferenciales del cálculo vectorial: el gradiente y el Laplaciano. En tres dimensiones, el vector gradiente propuesto por Koshizuka y Oka (1996), uno de los tres tipos de diferenciación de primer orden del cálculo vectorial (los otros dos son la divergencia y el rotacional), se representa por la siguiente ecuación de discretización para la partícula i:

$$\langle \nabla \phi \rangle_i = \frac{d}{n_i} \sum_{j \neq i} \frac{\phi_j - \phi_i}{\left|\vec{r}_j - \vec{r}_i\right|} \frac{\vec{r}_j - \vec{r}_i}{\left|\vec{r}_j - \vec{r}_i\right|} w\left(\left|\vec{r}_j - \vec{r}_i\right|\right), \quad (9).$$

Donde $\vec{r}_i = (x_i, y_i, z_i)$ es el vector de posición de la partícula i. $\frac{\phi_j - \phi_i}{|\vec{r}_j - \vec{r}_i|}$ es la magnitud del vector gradiente entre las partículas i y j, y $\frac{\vec{r}_j - \vec{r}_i}{|\vec{r}_j - \vec{r}_i|}$ es el vector unitario de la partícula i a la partícula j. d es el número de

dimensiones espaciales: $d = 2$ para dos dimensiones y $d = 3$ para tres dimensiones. $\frac{\phi_j - \phi_i}{|\vec{r_j} - \vec{r_i}|}\frac{\vec{r_j} - \vec{r_i}}{|\vec{r_j} - \vec{r_i}|}$ representa al vector gradiente entre las partículas *i* y *j*. El lado derecho de la ecuación (9) es el promedio pesado de los vectores gradiente entre la partícula *i* y sus partículas vecinas *j*, ver figura 2.3. El resultado de la suma se multiplica por la dimensión espacial d porque el vector gradiente calculado tiene la información unidimensional en la dirección de $\frac{\vec{r_j} - \vec{r_i}}{|\vec{r_j} - \vec{r_i}|}$.

Para un vector arbitrario $\vec{\phi}$, Tanaka y Masunaga (2010) propusieron la siguiente expresión para discretizar el operador de divergencia:

$$\langle \nabla \cdot \vec{\phi} \rangle_i = \frac{d}{n_i} \sum_{i \neq j} \left[\frac{(\overrightarrow{\phi_j} - \overrightarrow{\phi_i}) \cdot (\vec{r_j} - \vec{r_i})}{|\vec{r_j} - \vec{r_i}|^2} w(|\vec{r_j} - \vec{r_i}|) \right], \quad (10)$$

La divergencia del gradiente de una función escalar se llama Laplaciano. El operador Laplaciano propuesto por Koshizuka y Oka (1996), el cual es la diferenciación espacial de segundo orden y en física representa la difusión, se escribe como:

$$\langle \nabla^2 \phi \rangle_i = \frac{2d}{\lambda_i n_i} \sum_{j \neq i} (\phi_j - \phi_i)\, w(|\vec{r_j} - \vec{r_i}|), \quad (11)$$

donde

$$\lambda_i = \frac{\sum_{j \neq i} |r_j - r_i|^2 w(|r_j - r_i|)}{\sum_{j \neq i} w(|r_j - r_i|)}, \quad (12).$$

La ecuación (8) expresa que una parte de la cantidad ϕ de la partícula i se transfiere a la partícula j:

$$\langle \Delta \phi_{i \to j} \rangle_i = \frac{2d}{\lambda_i n_i} \phi_i w(|r_j - r_i|), \quad (13)$$

y de la misma manera una parte de la cantidad ϕ de la partícula j se transfiere a la partícula i:

$$\langle \Delta \phi_{j \to i} \rangle_i = \frac{2d}{\lambda_i n_i} \phi_j w(|r_j - r_i|), \quad (14)$$

donde $\langle\ \rangle_i$ es la evaluación en la partícula i. La partícula i pierde la cantidad representada por la ecuación (13) y gana la representada por la ecuación (14) como se muestra en el lado derecho de la ecuación (11). La cantidad obtenida por la partícula j es:

$$\langle \Delta \phi_{i \to j} \rangle_j = \frac{2d}{\lambda_j n_j} \phi_i w(|r_i - r_j|), \quad (15)$$

Si λ_i y n_i conservan los mismos valores para todas las partículas, es decir $\lambda_i = \lambda_j$ y $n_i = n_j$, las ecuaciones (13) y (15) son iguales y la cantidad perdida por i es exactamente la obtenida por j. En este caso, la cantidad se conserva localmente entre las partículas i y j.

La formulación de la discretización del operador Laplaciano en el método MPS representa la distribución de una cantidad de la partícula i hacia sus partículas vecinas j, ver figura 2.4. La distribución depende de la función de peso.

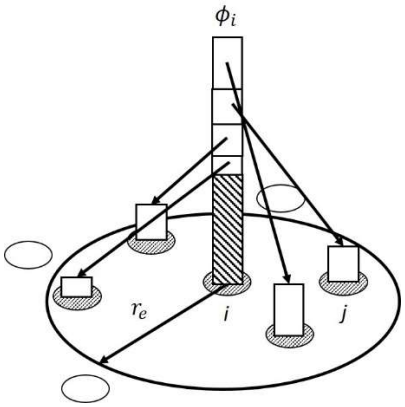


Figura 2.4. Modelo del Laplaciano.

2.5 Breve esbozo del método MPS.

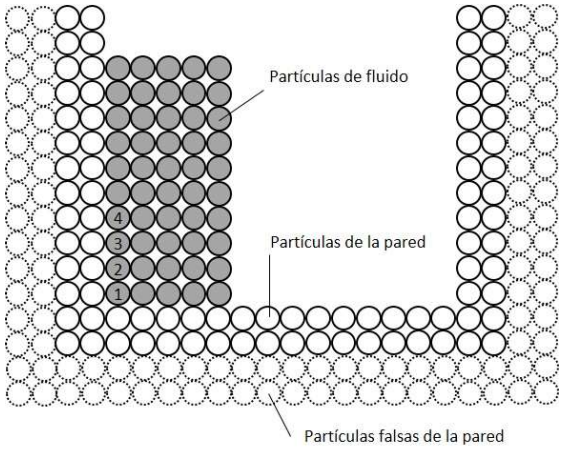
En esta sección se explica cómo se mueven las partículas usando el método MPS. Se comienza definiendo al fluido como un arreglo de partículas, la posición de las partículas representa la configuración inicial del fluido, en la figura 2.5 se muestra una columna de agua que está a punto de romperse por la acción de la gravedad.

Las partículas se colocan a intervalos regulares, a una distancia inicial l_0. En la figura 2.5a las partículas se representan como círculos, con superficies lisas. Sin embargo, las partículas de hecho son puntos en el método MPS, figura 2.5b, y por supuesto no poseen superficie. La distancia inicial entre las partículas, l_0, representa la resolución espacial del método MPS, es decir, valores pequeños de l_0 representan una resolución espacial alta, pero requieren un número mayor de partículas y de tiempo de cálculo.

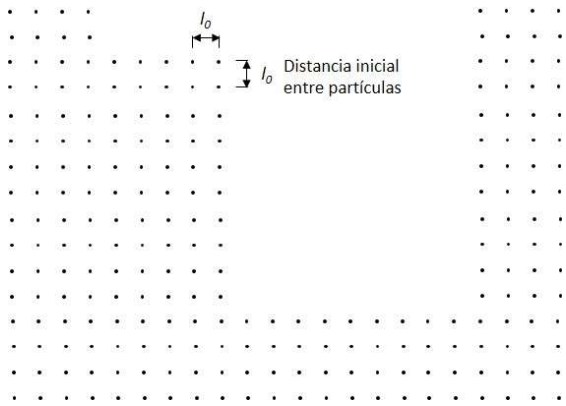
La figura 2.6 muestra un diagrama esquemático del movimiento de partículas para un paso de tiempo en la simulación del colapso de la columna de agua, también conocida como rompimiento de una presa. La figura 2.6a representa la distribución de las partículas en el estado inicial, observe que las partículas están colocadas en arreglos regulares. Las partículas se mueven por la aceleración que experimentan y que se puede calcular usando la ecuación de Navier-Stokes.

Para calcular la aceleración de cada partícula, el método semi-implicto se basa en el método predictor-corrector o de pasos fraccionados que es ampliamente usado en los métodos basado en mallas. En este esquema la ecuación de Navier-Stokes que se resuelve en dos pasos.

El primer paso se conoce como el paso *predictor* donde la contribución del gradiente de presión no se considera en el cálculo del movimiento de las partículas (fase explicita), lo que ocasiona la aparición de áreas en el fluido donde la densidad del número de partículas no es constante, es decir donde no se satisface la condición de incompresibilidad. En la figura 2.6b se muestra la fase justo después del desplazamiento de las partículas debido a la velocidad temporal calculada en la fase explicita, mismo que debe de ser corregido.



a) Partículas representadas como círculos.



b) Partículas representadas como puntos.

Figura 2.5. Arreglo de partículas y su representación.

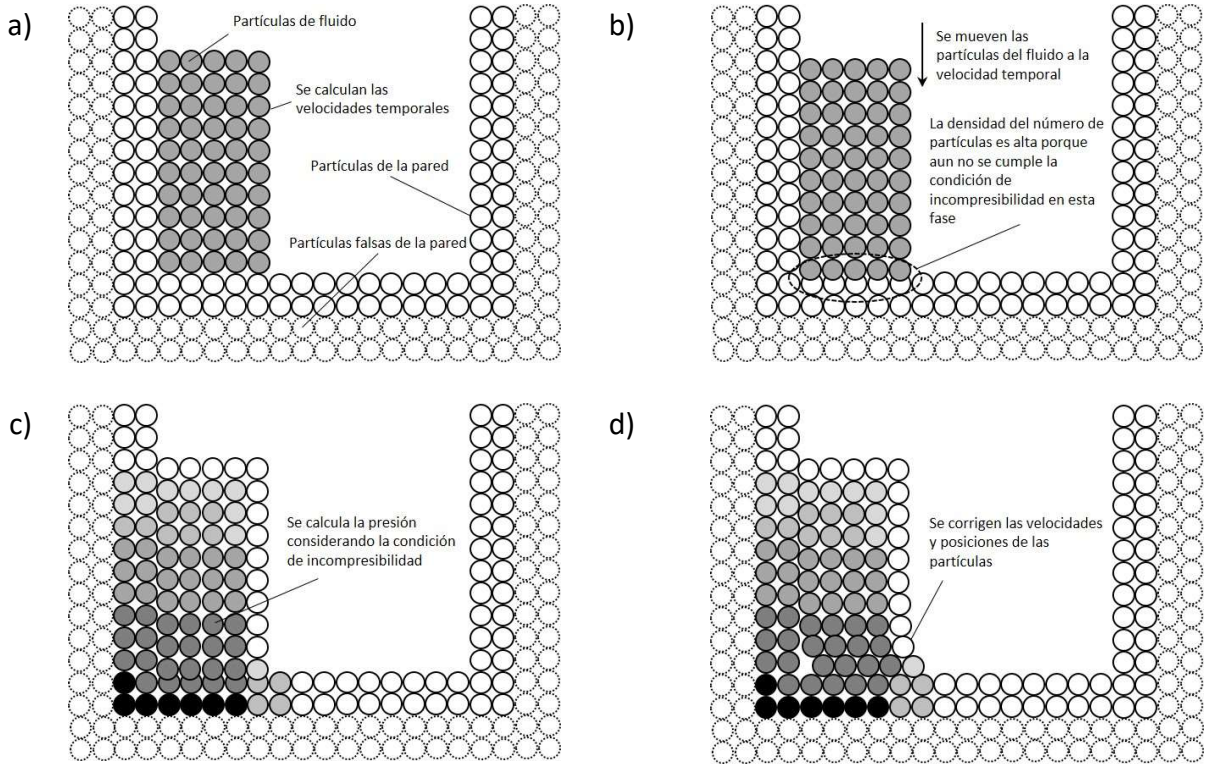


Figura 2.6. Diagrama esquemático del movimiento de las partículas en un paso de tiempo usando el algoritmo semi-implicito del método MPS. a) Fase inicial del paso de tiempo, en la cual se definen las condiciones iniciales del fluido, b) fase explicita justo después de mover las partículas usando las velocidades temporales sin considerar las fuerzas de la presión, c) fase implícita en la cual con la distribución de la presión de las partículas se calcula el gradiente de presión, y d) la fase después de corregir las velocidades y posiciones de las partículas usando el gradiente de presión.

2.5.1 Calculo de la distribución de presiones

Para corregir los desplazamientos se debe de considerar la aceleración debida a la presión, sin embargo, en la ecuación de Navier-Stokes el valor de la presión se desconoce, lo que hace inviable el cálculo de la aceleración de las partículas del fluido. Este problema se resuelve introduciendo la ecuación de continuidad.

Ahora bien, para satisfacer la condición de incompresibilidad y al mismo tiempo la ecuación de continuidad, el método MPS considera el término del gradiente de la presión como la contribución que compensa la diferencia entre la densidad de número de partículas. En este paso intermedio se formula la ecuación de Poisson para calcular el valor de la presión en cada partícula.

La ecuación de Poisson es una ecuación diferencial elíptica definida como $\nabla^2\phi = b$, donde ϕ es una cantidad arbitraria, b es el *término fuente* y es una constante o una función que no depende de ϕ. En el método MPS, el término fuente se calcula a partir de la desviación que existe entre la densidad del número de partículas temporal n^* y el valor inicial n^0. La densidad del número de partículas es proporcional a la densidad ρ. Entonces, el término fuente se refiere a la desviación de la densidad con respecto a un valor constante. El campo de presiones se evalúa implícitamente para conservar la densidad constante, lo que significa que la ecuación (2) se satisface para cumplir la condición de continuidad para un fluido incompresible. La ecuación de Poisson de la presión del método MPS es:

$$\langle \nabla^2 P \rangle_i^{k+1} = -\frac{\rho^0}{(\Delta t)^2}\left(\frac{n_i^* - n^0}{n^0}\right) \quad (16)$$

El lado derecho de la ecuación (16) es el término fuente, y depende de la densidad del número de partículas con lo que se busca obtener la compensación requerida para mantener constante la densidad del número de partículas en cada paso de la simulación. Cuando se comprime el fluido, la distancia entre las partículas difiere del valor inicial, este cambio se observa si se evalúa la función de peso de la densidad del número de partículas. En tanto, el lado izquierdo de la ecuación (16) se discretiza con el modelo del Laplaciano de la ecuación (11), con lo que se obtiene un sistema de ecuaciones cuyas incógnitas son las presiones en cada partícula.

De manera similar, el método del volumen finito, hace uso de una formulación parecida a la de la ecuación (16) para expresar el término fuente de la ecuación de presión de Poisson y se representa por la divergencia de la velocidad:

$$\langle \nabla^2 P \rangle_i^{k+1} = \rho^0 \frac{\langle \nabla \cdot \vec{u} \rangle_i^*}{\Delta t} \quad (17)$$

La figura 2.6c muestra la imagen conceptual de la distribución de presiones. A partir de la presión se calcula el gradiente de presión $-\frac{1}{\rho}\nabla P$ para corregir el campo de velocidad y la posición de las partículas (fase implícita). En la figura 2.6d se muestra la imagen conceptual de la distribución de la presión después de la corrección.

2.5.2 Condiciones de frontera para calcular la presión

Se usan dos tipos de condiciones de frontera. Un tipo es la condición de frontera de Dirichlet, que consiste en fijar valores a ciertos parámetros, en el método MPS el valor asignado es 0 Pa de presión a las partículas de la superficie libre. Existen algunas técnicas para juzgar si una partícula está en la superficie libre o no. La técnica tradicional usa la densidad del número de partículas propuesta por Koshizuka y Oka (1996). Si la densidad del número de partículas satisface la siguiente condición se considera como una partícula en la superficie libre y se le asigna una presión de 0 Pa.

$$n_i < \beta n^0 \quad (18)$$

donde β es un parámetro escalar cuyo valor se determina empíricamente y por lo general adopta el valor de 0.97.

El segundo tipo de condición de frontera es la condición de frontera de Neumann, descrita mediante primeras derivadas, y se usa para representar las paredes durante el cálculo de la presión. En el método MPS, la condición de frontera de Neumann consiste asignar un gradiente de presión cero a las paredes. Para representar de manera aproximada la condición de frontera de Neumann se colocan dos capas de partículas falsas, que no tienen valores de presión, detrás de una o dos capas de partículas que representan las paredes, y que si tienen valores de presión. Con la condición de frontera de Neumann se evita que las partículas de fluido penetren las paredes.

2.5.3 Condiciones de frontera para el cálculo de la velocidad

En dinámica de fluidos, la condición anti-deslizamiento de fluidos viscosos asume que en una frontera sólida, el fluido tendrá una velocidad cero relativa a la frontera. De aquí que, la condición de anti-deslizamiento de una pared fija se satisface al asignar a las partículas de la pared una velocidad de 0 m/s. Para conseguir lo anterior, se omiten las partículas de la pared vecinas del cálculo de la viscosidad de una partícula de fluido. Esto es, si la partícula vecina *j* es una partícula de la pared, no se calcula la interacción entre la partícula del fluido *i* y la partícula de la pared *j* porque las fuerzas de viscosidad no ejercen influencia entre el fluido y la frontera.

2.5.4 Detalles del algoritmo semi-implícito

En el método MPS, la ecuación de continuidad y el término gradiente de presión en la ecuación de Navier-Stokes se evalúan en cada paso de tiempo nuevo y los demás términos se evalúan en el paso de tiempo actual:

$$\left.\frac{D\rho}{Dt}\right|^{k+1} = 0, \quad (19)$$

$$\frac{D\vec{u}}{Dt} = \left.-\frac{1}{\rho}\nabla P\right|^{k+1} + \left.\frac{\mu}{\rho}\nabla^2\vec{u}\right|^{k} + \left.\frac{\vec{g}}{\rho}\right|^{k}, (20)$$

donde los superíndices k y $k+1$ representan los pasos de tiempo actual y nuevo respectivamente. Se resuelven las ecuaciones (19) y (20) en dos pasos. En el primer paso, los términos que incluyen la viscosidad y gravedad se calculan en el paso de tiempo actual. En el método MPS el término de la

viscosidad en la ecuación de Navier-Stokes (21) se discretiza usando la expresión (11) del Laplaciano y se calcula como:

$$\upsilon\langle\nabla^2\vec{u}\rangle_i = \upsilon\frac{2d}{\lambda_\upsilon n^0}\sum_{i\neq j}\left(\vec{u}_j-\vec{u}_i\right)w\left(\left|\vec{r}_j-\vec{r}_i\right|\right), \quad (22)$$

donde υ es el coeficiente de viscosidad cinematica y se define como $\upsilon = \mu/\rho$. El vector de velocidad se obtiene como:

$$\vec{u}^* = \vec{u}^k + \Delta t\left[\upsilon\nabla^2\vec{u} + \frac{\vec{g}}{\rho}\right]^k, \quad (22)$$

donde el superíndice * indica que se trata del valor intermedio, es decir, de la fase explícita del método. A partir de la velocidad es posible calcular el movimiento intermedio o virtual, como sigue:

$$\vec{r}^* = \vec{r}^k + \Delta t\vec{u}^*, \qquad (23)$$

Este es el paso explícito. De esta configuración virtual $\vec{r}^*$, la condición de incompresibilidad no se satisface $n^0 \neq \langle n\rangle_i = n_i$, entonces es necesaria una corrección.

En el segundo paso, el paso implícito, se calcula el nuevo valor $k\ +1$, de la presión con la ecuación (16) de Poisson considerando la incompresibilidad. Para calcular la ecuación de la presión de Poisson se aplica el modelo Laplaciano del método MPS al lado izquierdo de la ecuación (16) como sigue:

$$\frac{1}{\rho^0}\frac{2d}{\lambda^0 n^0}\sum_{j\neq i}\left(P_j^{k+1}-P_i^{k+1}\right)w\left(\left|\vec{r}_j^*-\vec{r}_i^*\right|\right) = -\frac{1}{(\Delta t)^2}\frac{n_i^*-n^0}{n^0}, \quad (24)$$

El vector de velocidad temporal se actualiza con el término del gradiente de presión en el nuevo paso de tiempo:

$$\vec{u}^{k+1} = \vec{u}^* - \frac{\Delta t}{\rho}\nabla P\big|^{k+1}, \quad (25)$$

Finalmente, se corrige la posición de las partículas de $\vec{r}^*$a $\vec{r}^{k+1}$ usando $\vec{u}^{k+1}$ como sigue:

$$\vec{r}^{k+1} = \vec{r}^* + (\vec{u}^{k+1}-\vec{u}^*)\Delta t, \quad (26).$$

2.6 Método MPS modificado

En este trabajo se usó el modelo del método MPS modificado por Sanchez-Mondragón (2016). Las modificaciones corrigen tres problemas observados en el método MPS original:

- Hace más eficiente la búsqueda de partículas que están al interior del fluido y que erróneamente se clasificaron como partículas de la superficie libre, con presión en cero, de manera que se estabiliza el campo de presiones durante las simulaciones.
- Estabiliza las oscilaciones en la presión al mismo tiempo reduce la amplitud y frecuencias altas durante el impacto violento de las partículas, para lo cual se modificó el parámetro de variancia del operador Laplaciano en la ecuación de presión de Poisson.
- Provee una definición más estable del término del vector del gradiente de presión.

A continuación, se muestran las formulas empleadas en el algoritmo modificado.

2.6.1 Función de peso

La función de peso, ecuación (2), se reemplaza por la propuesta por Lee et al. (2011):

$$w(r) = \begin{cases} \left(1 - \frac{r_e}{r_{ij}}\right)^3 \left(1 + \frac{r_e}{r_{ij}}\right)^3, & (0 \le r_{ij} < r_e) \\ 0, & (r_e < r_{ij}) \end{cases}, \quad (27)$$

donde r_{ij} es la distancia entre partículas $\vec{r}_i$ y $\vec{r}_j$ con $r_{ij} = \left|\vec{r}_i - \vec{r}_j\right|$ y r_e es el radio efectivo.

2.6.2 Gradiente de presión

El término del gradiente de presión usado en este trabajo es el modificado por Sanchez-Mondragón (2016), y se calcula con la siguiente ecuación:

$$-\left(\frac{1}{\rho}\langle \nabla P\rangle\right)_i = -\frac{d}{n^0}\sum_{i\neq j}\left[\frac{2}{\rho_i + \rho_j}\frac{P_j - \widehat{P}_i}{\left|\vec{r}_j - \vec{r}_i\right|^2}\left(\vec{r}_j - \vec{r}_i\right)w\left(\left|\vec{r}_j - \vec{r}_i\right|\right)\right] - \frac{d}{n^0}\left[\frac{2}{\rho_i + \rho_{j_{min}}}\frac{P_j - \widehat{P}_i}{\left|\vec{r}_{jmin} - \vec{r}_i\right|^2}\left(\vec{r}_{jmin} - \vec{r}_i\right)w\left(\left|\vec{r}_{jmin} - \vec{r}_i\right|\right)\right], \quad (28)$$

2.6.3 Ecuación de la presión de Poisson

El término fuente de la ecuación de Poisson se ha modificado para mejorar el cálculo de los valores de la presión, una de estas modificaciones es la propuesta por Tanaka y Masunaga (2010):

$$\langle \nabla^2 P\rangle_i = \frac{\rho}{\Delta t}\nabla \cdot \vec{u}^* + \gamma \frac{\rho}{\Delta t^2}\frac{n_i - n^0}{n^0}, \quad (29)$$

que posteriormente fue modificada por Lee et al. (2011) como sigue:

$$\langle \nabla^2 P\rangle_i = (1-\gamma)\frac{\rho}{\Delta t}\nabla \cdot \vec{u}^* + \gamma \frac{\rho}{\Delta t^2}\frac{n_i - n^0}{n^0}, \quad (30)$$

donde el parámetro γ es menor a 1.0 y en este trabajo su valor es de 0.25.

2.6.4 Parámetro de varianza

Sanchez-Mondragon (2016) sugirió una modificación para el parámetro de varianza, ecuación (9), el cual considera la contribución de varianza de la velocidad, este parámetro se escribe de la siguiente forma:

$$\lambda_i^* = \frac{r_0^2}{5} - \frac{\sum_{j\neq i} w\left(\left|\vec{r}_j - \vec{r}_i\right|\right)\left(\vec{u}_j - \vec{u}_i\right)^2 (\Delta t)^2}{\sum_{j\neq i} w\left(\left|\vec{r}_j - \vec{r}_i\right|\right)}, \quad (31)$$

La definición de λ_i^* de la ecuación (31) se aplica a la expresión propuesta por Lee et al. (2011):

$$\sum_{i\neq j}(P_j - P_i)\, w(|\vec{r}_j - \vec{r}_i|)$$

$$= (1-\gamma)\frac{\lambda_i^* \rho_i}{2\Delta t}\sum_{i\neq j}\left[\frac{(\vec{u}_j^* - \vec{u}_i^*)\cdot(\vec{r}_j - \vec{r}_i)}{|\vec{r}_j - \vec{r}_i|^2} w(|\vec{r}_j - \vec{r}_i|)\right] + \gamma\frac{\lambda_i^* \rho_i}{2\Delta t^2}(n^0 - n_i^k), \quad (32)$$

Entonces, cuando la velocidad virtual $\vec{u}^*$ presenta incrementos grandes $\vec{u}_j^* - \vec{u}_i^*$, el parámetro virtual λ_i^* compensara la divergencia del modelo disminuyendo.

Para resolver la ecuación de Poisson de la presión se usan las condiciones de frontera de Dirichlet, para lo cual en el método MPS se asigna el valor de la presión atmosférica cero a las partículas que se encuentran en la superficie libre.

Las partículas en la superficie libre se definen a través de la densidad del número de partículas, si satisfacen la siguiente ecuación:

$$\langle n \rangle_i < \beta n^0, \quad (33)$$

Con β entre 0.8 y 0.99, Khayer y Gotoh (2009), los valores más usados $\beta = 0.95$ y $\beta = 0.97$.

La búsqueda de partículas en la superficie libre se realiza usando el método propuesto por Sanchez-Modragon (2016); para una constante β entre 0.8 y 0.99, y otra constante β_1 ($\beta_1 < \beta$). Las partículas i con una densidad de número de partículas mayor que βn^0 no están en la superficie libre, pero las partículas i que se encuentran en la zona de dispersión:

$$\beta_1 n^0 \leq n_i \leq \beta n^0, \quad (34)$$

Para definir si i es una partícula de la superficie libre, se consideran la densidad de número de partículas de sus vecinos j, entonces si se cumple la siguiente condición:

$$\beta_1 n^0 \leq n_j, \forall\ vecinos\ j, \quad (35)$$

Entonces la partícula i no está en la superficie libre.

2.6.5 Modelo de colisión

El modelo de colisión consiste en detectar partículas que se encuentran a una distancia menor que a r^0, con a un parámetro de proporción y r^0 la distancia inicial entre partículas. Entonces, es aplicada la conservación de momento y la velocidad de repulsión es calculada como:

$$\vec{v}' = -b\vec{v}, \quad (36)$$

donde $\vec{v}'$es la velocidad resultante, *b* es el coeficiente de colisión y $\vec{v}$ es la velocidad inicial. Los valores propuestos por Lee et al. 2011 son: a = 0.5 y b = 0.2.

3 Programación serial del método MPS.

En esta sección se presenta la programación serial en lenguaje C del método MPS. El programa está basado en las rutinas construidas por Sanchez-Mondragon (2016) en MatLab. Para comenzar primero se discutirá el diagrama de flujo del método MPS mostrado en la figura 3.1.

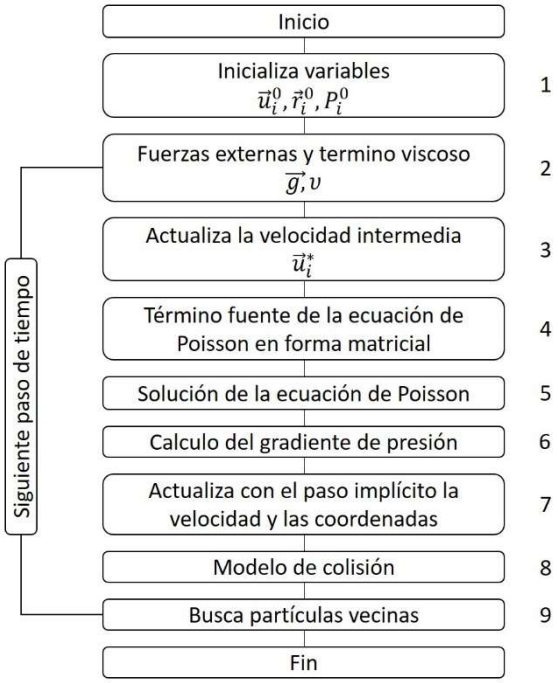


Figura 3.1. Diagrama de bloques del algoritmo del método MPS.

En el paso 1 se inicializa las posiciones, velocidades y presiones de las partículas. En el siguiente paso se calculan las fuerzas de gravedad y las de viscosidad. En el paso 3 se calcula la velocidad intermedia sin considerar el término del gradiente de presión, ecuación (22). El término fuente de la ecuación (16) se calcula en el paso 4. En el paso 4 también se expande el Laplaciano de la presión como indica la ecuación (24), donde P_i y P_j son incógnitas y forman un arreglo de ecuaciones simultáneas.

En el paso 5 se resuelve el sistema de ecuaciones simultáneas, este sistema es simétrico, bandeado y definido positivo, pero cuya representación matricial es la de una matriz dispersa, lo que da pie al uso de algún método iterativo, como el método del gradiente conjugado. Con las presiones calculadas en las partículas del fluido que no están en la superficie libre, en el paso 6 se calculan los gradientes de presión usando la ecuación (28). El gradiente de presión es el término que faltaba para actualizar la posición y velocidad de cada partícula, ecuación (26), ésta se hace en el paso 7. En el paso 8 se corrige, con el modelo de colisión, la posición y velocidad de las partículas que atraviesen las paredes del contenedor. Finalmente, se actualiza la lista de vecinos de cada partícula en el paso 9. Los pasos 2 a 9 se repiten sucesivamente para los siguientes instantes tiempos hasta completar el tiempo total de análisis.

Previo a la programación serial del método MPS, se explicarán brevemente dos algoritmos que son importantes cuando se trata de mejorar el tiempo de ejecución: el algoritmo de búsqueda de partículas vecinas y el algoritmo del gradiente conjugado.

3.1 Algoritmo de búsqueda de partículas vecinas

La búsqueda de las partículas que interactúan es un paso muy importante en el método MPS es por eso que se dedica esta sección a explicar el algoritmo empleado. Como se recordará de la sección 1.2, la interacción entre partículas solo ocurre dentro de un radio efectivo. La forma más simple de identificar

las partículas vecinas es el algoritmo de búsqueda exhaustivo que analiza todos los pares de partículas. Propiamente este algoritmo no busca las parejas que interactúan, sino que obtiene todas las interacciones potenciales entre partículas. Debido a que este algoritmo se aplica para las N partículas del fluido, y que para cada partícula se calcula N veces la distancia entre partículas, es claro que la complejidad de este algoritmo es del orden de $O(N^2)$. Además, el proceso de búsqueda se repite en cada paso de tiempo, esto representa un alto costo de tiempo de procesamiento.

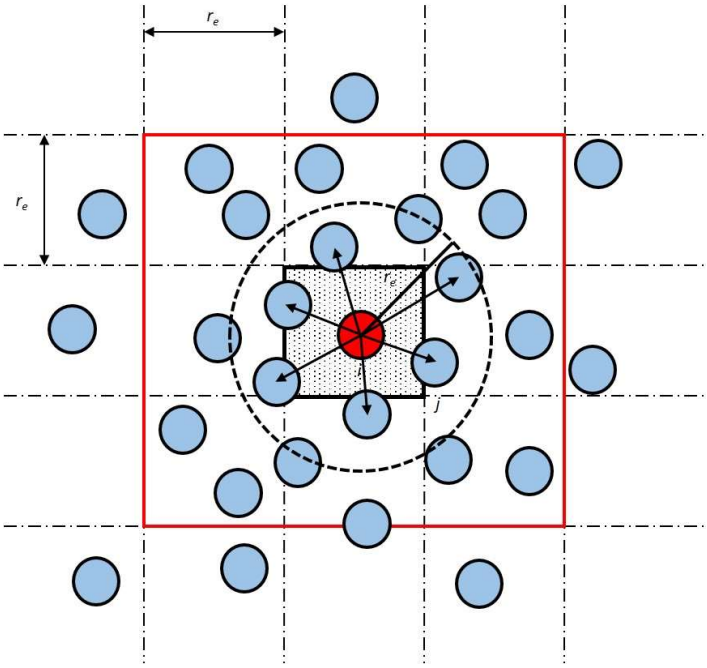


Figura 3.2. Malla adicional en dos dimensiones usada en el algoritmo de búsqueda de listas ligadas para encontrar partículas vecinas.

Una forma más efectiva de encontrar las partículas vecinas con las que realmente se interactúa es emplear el enfoque de búsqueda de listas ligadas. La idea principal de este enfoque es usar una malla adicional, y se ha usado en el método de partículas SPH, y fue propuesto originalmente por Monaghan y Lattanzio (1985). De cierta manera, éste es un enfoque partícula-malla, el cual difiere del método sin mallas MPS. La malla divide el dominio en celdas con las siguientes características:

- se usan para definir la distribución espacial de las partículas, y así identificar las partículas vecinas,
- y tienen una geometría cartesiana simple, misma que no se ajusta al dominio computacional.

El algoritmo para identificar a las partículas vecinas con una malla adicional trabaja como sigue. Primero, se sobrepone una malla cartesiana simple sobre el dominio computacional. Esta malla debe ser lo suficientemente grande para considerar aquellas partículas que salen del contenedor debido al chapoteo. El espaciamiento de la malla depende del radio efectivo. La malla adicional, una vez creada, permanece sin cambios durante todo el tiempo que dure la simulación. Para cada partícula se determina la celda en que está ubicada en cada paso de tiempo. La lista resultante de partículas en cada celda se almacena en memoria. Entonces, para una partícula dada, solo se evalúan las distancias a las partículas localizadas en la misma celda o en las celdas adyacentes. En la figura 3.2, se muestra el algoritmo de búsqueda de listas

ligadas para un problema de dos dimensiones con 9 celdas donde se hace la búsqueda. El cuadrado, con aristas de color rojo, delimita las celdas donde es posible encontrar partículas que interactúan con la partícula en cuestión (roja). Solo las partículas dentro del radio de influencia son consideradas (las parejas de interacción se marcan con flechas).

3.2 Algoritmo del gradiente conjugado

El contexto en que se aplica el algoritmo del gradiente conjugado es la solución de un sistema de ecuaciones lineales $Ax = b$, donde A se llama *matriz de coeficientes*, b es el *vector del lado derecho* y x es el vector de soluciones. Formalmente la solución x se obtiene como $x = A^{-1}b$. La meta de cualquier método iterativo es producir una solución aproximada que este dentro de una tolerancia dada en un número pequeño de iteraciones u operaciones. En la mayoría de los casos, reducir la cantidad de iteraciones es por mucho la forma más eficiente de lograr un alto rendimiento. El algoritmo 3.1, muestra el método del gradiente conjugado. Se ha demostrado que este método converge a la mejor aproximación posible si la matriz de coeficientes es definida positiva y simétrica.

Algoritmo 3.1. Método del gradiente conjugado

Dada una solución inicial x_0, calcular $r_0 = b - Ax_0$ y asignar $p_0 = r_0$

For $k = 0,1, \ldots$

(1) Calcular y almacenar Ap_k

(2) Calcular $< r_k, r_k >$

(3) Calcular $< p_k, Ap_k >$

(4) $$\alpha_k = \frac{< r_k, r_k >}{< p_k, Ap_k >}$$

(5) $x_{k+1} = x_k + \alpha_k p_k$

(6) Calcular $r_{k+1} = r_k - \alpha_k Ap_k$

(7) Calcular $< r_{k+1}, r_{k+1} >$

(8) $$\beta_k = \frac{< r_{k+1}, r_{k+1} >}{< r_k, r_k >}$$

(9) Calcular $p_{k+1} = r_{k+1} + \beta_k p_k$

(10) $r_k = r_{k+1}$

3.3 Contenidos del programa serial.

El programa de cómputo del método MPS está organizado en funciones, el significado de las funciones, las constantes y las variables usadas se explican en las tablas 3.1, 3.2 y 3.3 respectivamente.

Tabla 3.1. Lista de funciones.

Nombre de la función	Argumentos	Tipo devuelto	Funcionalidad
inicubico	`double st1, double st2, double st3, double st4, double h, double hh, double hw, double **ma`	`void`	Genera los arreglos de las partículas: fluido, paredes y falsas.
srch	`double **ma, int fnn, int fn, double ra, int *cnt, int **nhd, double *den`	`void`	Encuentra las partículas vecinas usando el algoritmo de búsqueda de listas ligadas.

Nombre de la función	Argumentos	Tipo devuelto	Funcionalidad
flamx	`doublé **ma, int fnn, double ra, double **veo, double dt, int *cnt, int **nhd, double *den, double *lamx`	`void`	Calcula el parámetro de varianza λ_i^*de la ecuación (32).
ffree	`double *den, double beta, double no, double dt, double *roo, int *cnt, int **nhd, double *lamx, double lam, double **ma, double ra, double **veo, int fnn, int fn, double **prex, double *nsum, int *nfr, int *pz`	`void`	Construye la vector del sistema de ecuaciones para el cálculo de la presión en cada partícula bajo las condiciones de frontera de Dirichlet.
fdivv	`double **ma, double ra, double **veo, double no, int *cnt, int **nhd, int elem`	`double`	Calcula el divergente de la velocidad.
fprex	`double **ma, double **prex, int *nfr, int *cnt, int **nhd, double ra, int nx`		Construye la matriz del sistema de ecuaciones para el cálculo de la presión en cada partícula.
gradx	`double **A, double *b, int n`	`double *`	Resuelve el sistema de ecuaciones usando el algoritmo del gradiente conjugado.
grphpres	`double **ma, int *pz, int su, double *pres, double pmax, int pesp, int fn, int in, int ke`	`void`	Genera un archivo que usa un código de colores para representar el rango de presión en cada partícula.
srchcol	`double **ma, int fnn, int fn, double ra, int *cnt, int **nhd, double *den, int in, double h, double **veo, double col`	`void`	Encuentra las partículas vecinas usando el algoritmo de búsqueda de listas ligadas, y corrige la posición y velocidad de las partículas que atraviesan las paredes usando el modelo de colisión.
main	`int argc, char **argv`	`int`	Lleva el control del método MPS.

Tabla 3.2. Lista de constantes simbólicas.

Constantes simbólica	Valor	Significado	Unidad
a	0.72	Ancho de la columna de agua.	m
l	0.36	Alto de la columna de agua.	m

Constantes simbólica	Valor	Significado	Unidad
bxa	1.932	Ancho de la pared inferior interna del contenedor.	m
bxl	0.72	Alto de las paredes verticales internas del contenedor.	m
h	0.012	Separación inicial entre las partículas.	m
beta	0.97	Constante para determinar si una partícula está en la superficie libre en función de la densidad del número de partículas.	-
no	2.50533	Densidad del número de partículas en el estado inicial.	-
dt	0.0014	Incremento del paso de tiempo.	s
nu	1.023e	Viscosidad dinámica.	m^2/s
rhof	1000	Densidad del fluido.	kg/m^3
rhow	1025	Densidad de las paredes.	kg/m^3
ra	2.1*h	Radio efectivo.	m
lam	(1/5)*ra^2	Lambda en la condición incial.	m^2
d	2	Dimensiones del modelo.	-
col	0.25	Coeficiente de colisión.	-
g	9.81	Aceleración de la gravedad.	m/s^2

Tabla 3.3. Lista de variables.

Variable	Tipo	Significado de la variable	Unidad
ma	`double **`	Posición (x,y) de las partículas.	m
veo	`double **`	Vector de velocidad (vx,vy) de las partículas.	m/s
vvis	`double **`	Vector de velocidad (vx,vy) debido a la viscosidad de las partículas.	m/s
sx, sy, r	`double`	Distancia en la dirección **x**, **y** y total entre la partículas i y j.	m
grpres	`double **`	Gradiente de presión en la dirección **x** y **y** de las partículas.	-
lamx	`double *`	Parámetro de varianza λ_i^*	-
cnt	`int *`	Contador de los vecinos de cada partícula.	-
nhd	`int **`	Lista de partículas vecinas de la partícula i.	-
den	`double *`	Densidad del número de partículas de la partícula i.	-
prex	`double **`	Matriz del sistema de ecuaciones de la ecuación de Poisson.	-
nsum	`double *`	Divergencia de la densidad del número de partículas para cada partícula.	-
pz	`int *`	Lista de partículas que están en la superficie libre del fluido.	-
su	`int`	Contador de partículas en la superficie libre del fluido	-
nfr	`int *`	Nuevo índice de las partículas en la lista auxiliar de partículas que exceptúa las partículas en la superficie libre del fluido.	-
pres1, pres2	`double *`	Lista de presiones en las partículas, la primera lista es parcial y la segunda se complementa con las partículas de la superficie libre.	kg/m^2

Variable	Tipo	Significado de la variable	Unidad
pmin, pmax	`double`	Presión mínima y presión máxima.	kg/m^2

3.4 Caso de estudio.

El caso de estudio consiste en simular el rompimiento de una presa, donde una columna de agua se colapsa debido a la gravedad. Las dimensiones del contenedor y de la columna de agua se muestran en la figura 3.3. Todo el modelo está constituido por 2956 partículas, con una separación inicial de 0.012m en ambas direcciones.

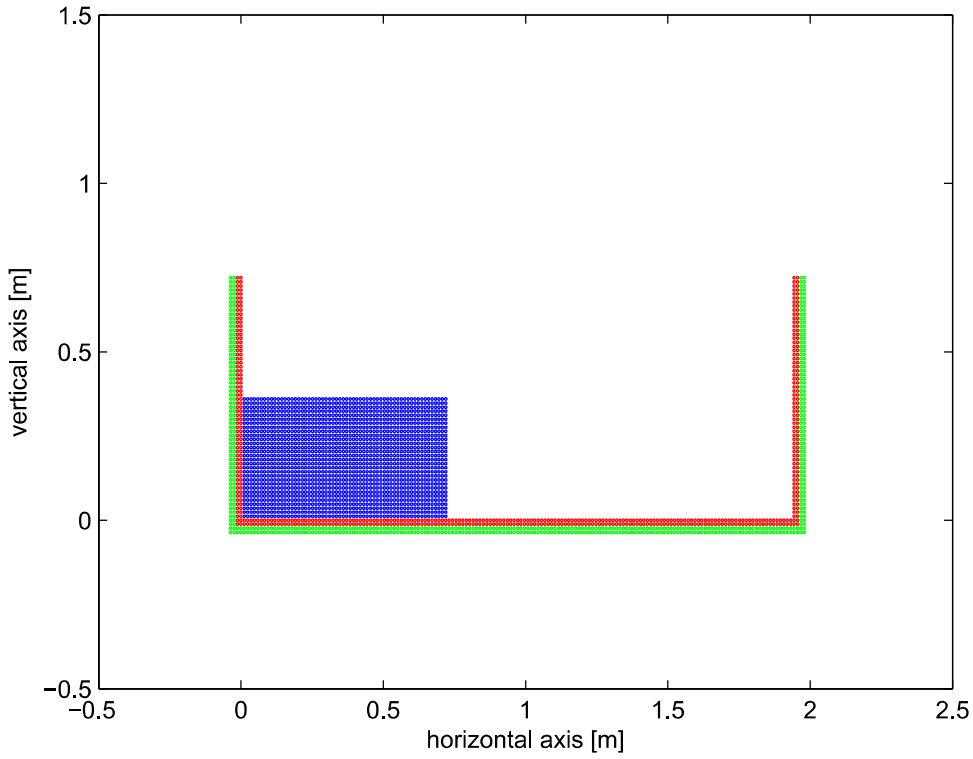


Figura 3.3. Diagrama esquemático del caso de estudio.

Las partículas de la columna del fluido son 1800 y están en color azul, se encuentran arregladas en una malla de 60 x 30 partículas. La partícula inferior izquierda está ubicada en la coordenada (0.012, 0.012). La densidad del fluido ρ = 1000 kg/m^3, y la constante cinemática de la viscosidad υ = 1.023 x 10^{-6} m^2/s. Las partículas del contenedor están en color rojo. El piso del contenedor está formado por dos capas horizontales cada una con 165 partículas; la primera capa comienza en la coordenada (-0.012, 0) y termina en la coordenada (1.956, 0); la segunda capa está debajo de la primera. Las paredes del contenedor están formadas por dos capas verticales cada una con 60 partículas; la primera capa de la pared izquierda comienza en la coordenada (0, 0.012) y termina en la coordenada (0, 0.72), la segunda capa está a la izquierda de la primera. Lo mismo sucede con la pared derecha, la primera capa comienza en la coordenada (1.944, 0.012) y termina en la coordenada (1.944, 0.72); la segunda capa está a la

derecha de la primera. Finalmente, las partículas falsas están en color verde y son un total de 586 distribuidas en dos capas alrededor del contenedor.

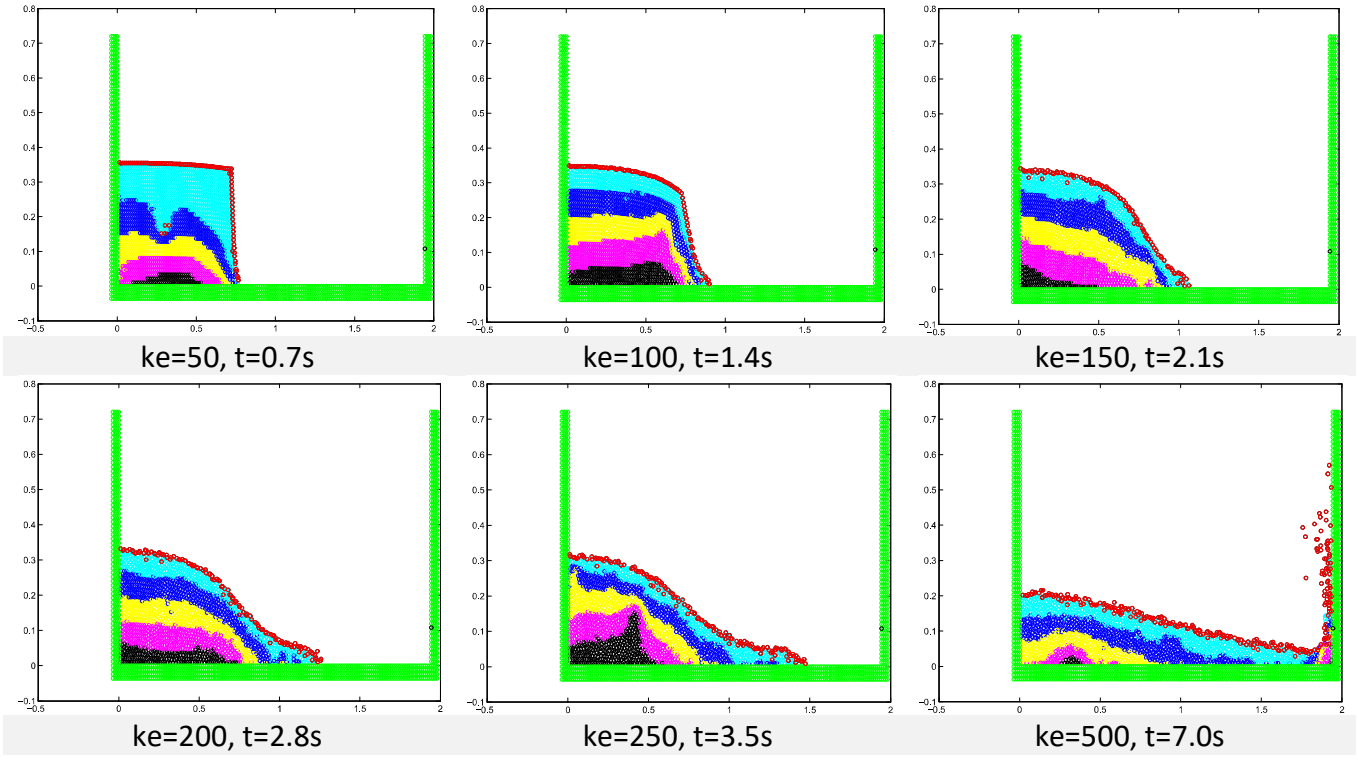
En el ejemplo se simulan 35 segundos del movimiento de las partículas con incrementos de 0.0014 segundos, es decir 2500 pasos. Se usó una computadora Dell Optiplex780 con procesador Core2 Quad 2.66GHz y memoria RAM de 8GB.

3.5 Resultados

El programa grafica la distribución de las partículas del fluido cada 50 pasos, "ke" es el contador que se incrementa cada 50 pasos, así como la presión normalizada (con respecto a la presión máxima) en cada partícula. En la figura 3.4 se presentan las gráficas correspondientes a 0.7s, 1.4s, 2.1s, 2.8s, 3.5s, 7.0s, 10.5s, 14.0s, 17.5s, 21.0s, 24.5s, 28.0s, 31.5s y 35.0s. Observe que en 7.0s y 10.5s algunas partículas están a punto de salir del contenedor después de chocar contra la pared derecha. La tabla 3.4 muestra el código de colores para representar la presión normalizada.

Presión norm.	0	0 < pres ≤ 0.2	0.2 < pres ≤ 0.4	0.4 < pres ≤ 0.6	0.6 < pres ≤ 0.8	0.8 < pres ≤ 1.0
Color						

Tabla 3.4. Códigos de colores para representar los rangos de presión normalizada.



ke=50, t=0.7s | ke=100, t=1.4s | ke=150, t=2.1s

ke=200, t=2.8s | ke=250, t=3.5s | ke=500, t=7.0s

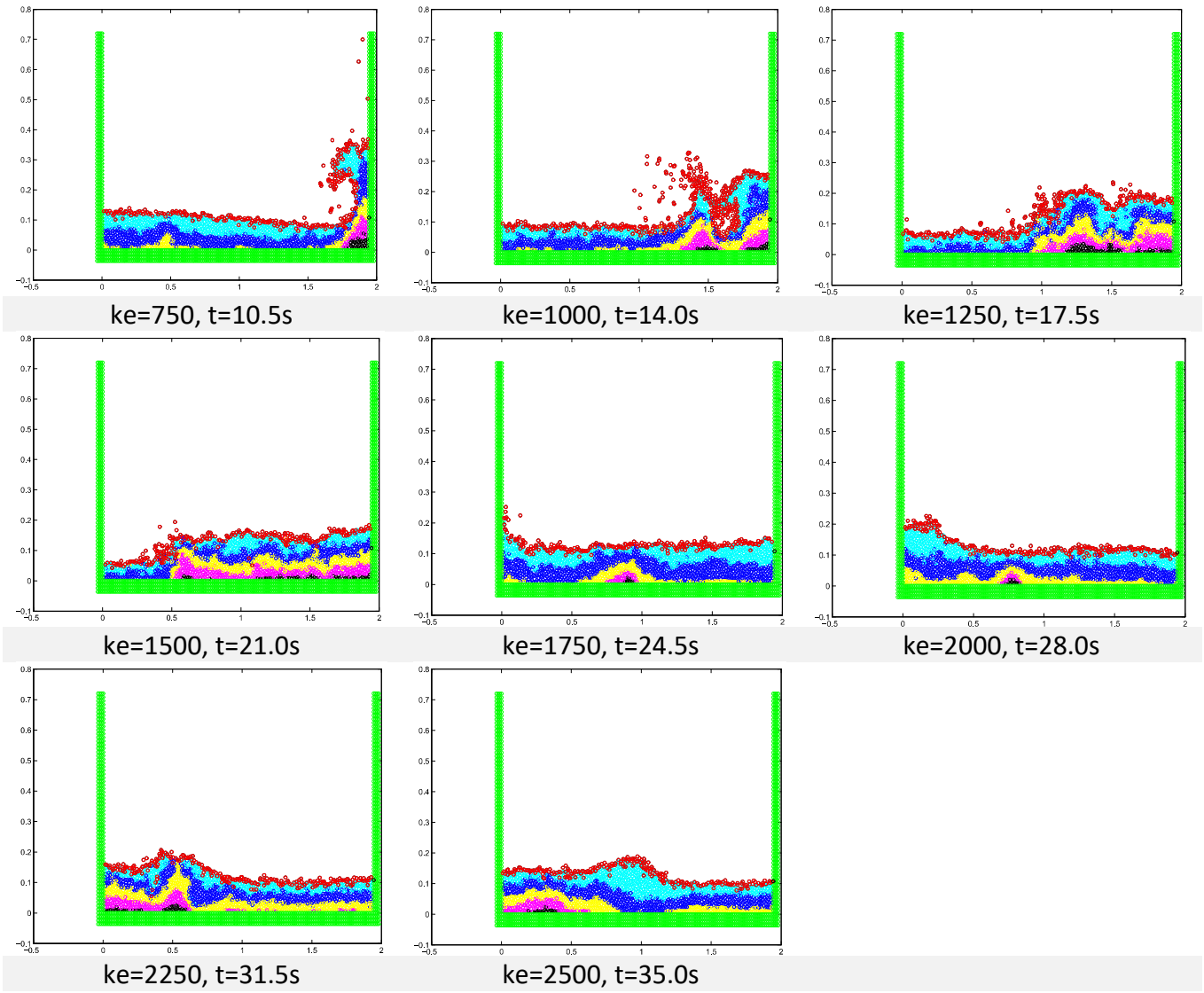


Figura 3.4. Simulación de la presión hidrodinámica normalizada.

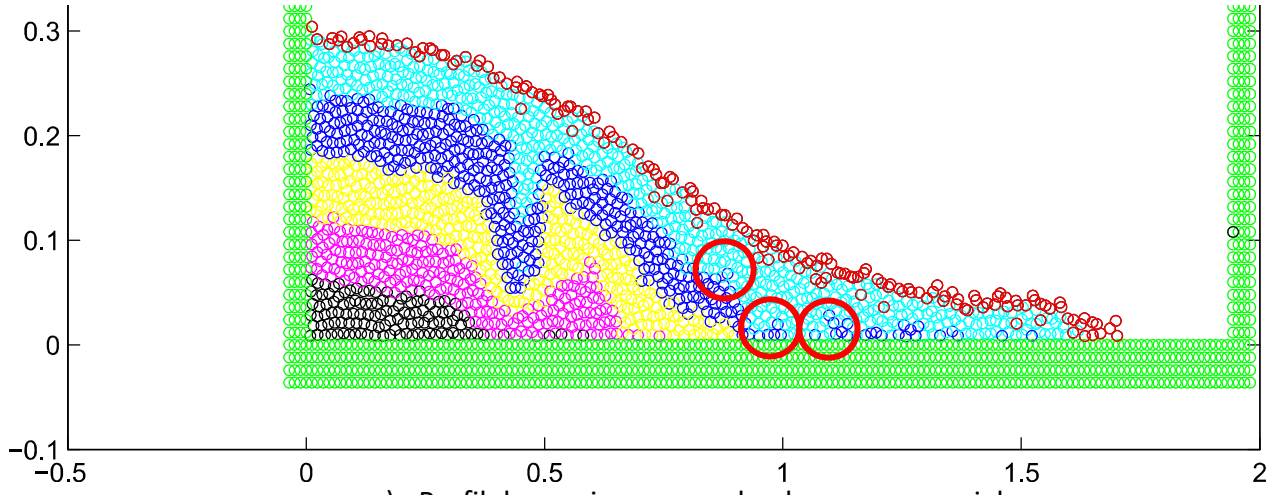
Se compararon las gráficas obtenidas contra las generadas por las rutinas del programa original en MatLab. Ambos conjuntos de graficas son idénticos hasta el paso 298, después de este paso hay cambios en los perfiles de presión, estos se deben a la sensibilidad del algoritmo para clasificar si una partícula está en la superficie libre o no, ecuaciones (34) y (35), y a mejoras que se hicieron en la programación del algoritmo del gradiente conjugado.

En la tabla 3.5 se muestran los resultados obtenidos en el paso 299 para la partícula 1553 usando MatLab y el programa serial en C. Las velocidades (veo_x y veo_y) y la posición de la partícula (ma_x y ma_y) son similares en ambos programas. Sin embargo, en el programa serial, a diferencia del programa de MatLab, el valor calculado de la densidad del número de partículas está en el rango definido en la ecuación (34) para considerar que la partícula 1553 está dentro del fluido. Lo anterior por supuesto influye en el gradiente de presión ($grpres_x$ y $grpres_y$) y en la presión (pres1) actuante en la partícula. La diferencia anterior modifica el perfil de presiones de los pasos subsecuentes.

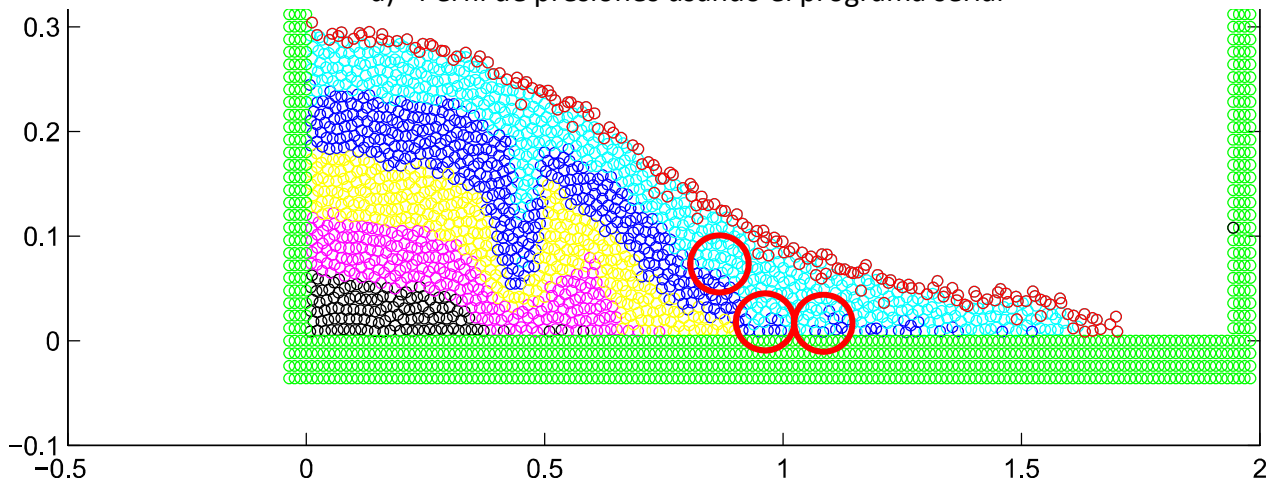
	veo_x	veo_y	ma_x	ma_y	$grpres_x$	$grpres_y$	pres1	prex
Serial	1.554545	-0.28714536	0.92379243	0.08001827	0.02516864	0.01559003	-505.208811	4981.25676
MatLab	1.543986072	-0.290706539	0.923762734	0.080008754	0.0146014	0.012042462	0	0

Tabla 3.5. Diferencias entre el método serial y el programa en MatLab.

En la figura 3.5 se observa que los perfiles de presiones en el paso 300 son similares con excepción de tres partículas ubicadas dentro los círculos rojos. Las partículas están en la misma posición, pero la presión calculada es diferente. Esto se debe a que tanto la posición como la velocidad de cada partícula dependen de los valores calculados en el paso anterior, pero la presión se calcula en cada paso resolviendo un sistema de ecuaciones cuyo tamaño es sensible a los valores de la densidad de partículas; estas diferencias se propagan en los siguientes pasos. Al término de la simulación se obtuvo un tiempo de ejecución de 10347 segundos.



a) Perfil de presiones usando el programa serial



b) Perfil de presiones usando MatLab.

Figura 3.5. Comparación de perfiles de presión.

4 Paralelización del método MPS

Históricamente en el mundo de la computación, la herramienta por excelencia para acelerar los cálculos es el paralelismo. El concepto clásico de un sistema de cómputo paralelo es un sistema en donde varios procesadores están conectados para que colaboren juntos en el mismo problema. Teóricamente, si un

programa emplea T segundos para completar una tarea, la misma puede terminarse en $\frac{T}{p}$ segundos si se usan p procesadores. Lo anterior llevaría a suponer que si se continúan agregando procesadores infinitamente, el problema se podría resolver sin consumir tiempo alguno. Esa *escalabilidad* perfecta es solo posible si el algoritmo también es escalable, es decir es completamente paralelizable, además si es posible escribir un programa eficiente y se dispone de un sistema de cómputo escalable que cuente con un número suficientemente grande de procesadores. Por lo que, el rendimiento altamente paralelo depende de tres factores, el sistema de cómputo, el algoritmo paralelo y el modelo de programación usado.

4.1 Arquitecturas de memoria compartida

Recordemos que este trabajo discute la paralelización del método MPS para un tipo de arquitectura paralela importante, la arquitectura de memoria compartida. Las arquitecturas de memoria compartida se pueden ver como la continuación de las arquitecturas de un solo procesador, donde todos los componentes de procesamiento tienen acceso una memoria de manera compartida. La tendencia en los últimos años en lo que se refiere a la arquitectura de microprocesadores indica que los sistemas de memoria compartida están alojados en un solo circuito. Los circuitos con múltiples procesadores se construyen colocando varios microprocesadores o núcleos en un solo circuito que trabajan en un espacio común de direcciones de memoria física. Aunque para el programador es transparente en cuanto a la funcionalidad, existen dos variedades de sistemas de memoria compartida cuyo rendimiento difiere en términos del acceso a la memoria principal:

- Sistema de acceso uniforme de memoria (UMA, Uniform Memory Access) que muestran un modelo de memoria “plano”, esto es, tanto la latencia como el ancho de banda es el mismo para todos los procesadores y todas las localidades de memoria. También se conoce como multiprocesamiento simétrico (SMP, Symmetric Multiprocessing). Los circuitos con múltiples procesadores son máquinas UMA.
- Sistemas con acceso de memoria no uniforme y coherencia de cache (ccNUMA cache-coherence Nonuniform Memory Access) cuya memoria está distribuida físicamente pero es compartida lógicamente. La capa física de estos sistemas es muy similar a la de los sistemas de memoria distribuida, pero la red lógica hace que la memoria agregada de todo el sistema se vea como un solo espacio de direcciones. La naturaleza distribuida influye en la rapidez con la que se tiene acceso a la memoria y varía dependiendo de la parte de la memoria a la que el procesador quiere tener acceso: local o remota.

Paralelizar un programa serial y producir código eficiente no es una tarea trivial. El proceso de poder ser dividido en cuatro pasos según Culler (1998):

1. *Descomposición* de los cálculos en tareas.
2. *Asignación* de las tareas a los hilos.
3. *Orquestación* de la sincronización necesaria entre los hilos.
4. *Mapear* o ligar los hilos con los procesadores.

En este trabajo se describe como se llevaron a cabo estos cuatro pasos en el paradigma de memoria compartida. Se continuará la revisión de las arquitecturas de memoria compartida con una introducción acerca de cómo se programan. Enseguida se analizará el programa serial para descomponerlo en tareas paralelas (descomposición) y se discutirán la sobrecarga por paralelismo asociada (asignación).

4.2 Introducción a la programación de memoria compartida con OpenMP.

La abstracción fundamental en el modelo de programación de memoria compartida es el concepto de *hilo*. Cuando se carga un programa, el sistema operativo prepara un espacio de direcciones para la aplicación que contiene el código del programa y sus datos. Para ejecutar el programa, el sistema operativo carga el inicio del programa en el contador del programa de un procesador. Después, ya que el espacio de direcciones se comparte y se disponen de varios procesadores, se carga el contador de programa de otro procesador para iniciar otro *hilo de control*. Los dos hilos se ejecutan en paralelo y se comunican leyendo y escribiendo en la memoria compartida.

Desde el punto de vista del programador, se crea un nuevo hilo al invocar una función de las bibliotecas del sistema y enseguida pasar la dirección de la dirección de una función donde los nuevos hilos comenzaran su ejecución. El cálculo en paralelo se obtiene si varios hilos se crean de esta manera y cada uno trabaja con datos diferentes. Aunque se mantiene la coherencia del espacio de direcciones, con frecuencia los hilos de una aplicación requieren ser sincronizados para asegurar la integridad de los datos.

4.2.1 OpenMP

El modelo de programación dominante de memoria compartida en cómputo científico es OpenMP. En OpenMP el hilo maestro se ejecuta hasta que se encuentra una región paralela. En este punto se crean varios hilos y entonces se divide el trabajo contenido en la sección paralela entre los hilos. Este estilo de programación frecuentemente se refiere como programación *fork/join*. La forma más simple de división del trabajo es un ciclo. Si las iteraciones dentro del ciclo son independientes, el trabajo se puede dividir asignando a un conjunto disjunto de indicies del ciclo a los diferentes hilos.

En OpenMP, las regiones paralelas se identifican con *directivas* o *pragmas* del compilador que se insertan en el código serial. Este método tiene la ventaja de que el código puede ser paralelizado de manera incremental conforme se añaden directivas. Además, las directivas se pueden ignorar si es necesario simplemente colocando una bandera del compilador lo que resulta en un programa serial.

4.3 Identificación del paralelismo en el método MPS

Para identificar que regiones de código son susceptibles a ser paralelizadas, se usó la herramienta de perfilado gprof del paquete *binutils* GNU. En la figura 4.1 se muestra un fragmento de la información sobre el tiempo de ejecución de todas las funciones que contiene el programa serial y la frecuencia de invocación.

De la salida del perfilador se observa que la función que consume más tiempo es *multAb* que multiplica una matriz A por un vector b y se usa tanto en el algoritmo del gradiente conjugado para encontrar la solución del sistema de ecuaciones como en el método MPS. La función se invocó 337,078 veces durante los 2500 pasos y consume el 94% (82.61% + 7.60% + 3.79%) del tiempo total de cómputo. La función *zeros_matriz* que inicializa una matriz con ceros se invocó 7501 veces y consumió el 1.23% del tiempo

total. De lo anterior, y para reducir el tiempo de ejecución, en un inicio la paralelización del método MPS se enfocó al algoritmo del gradiente conjugado.

Cada una de las operaciones del algoritmo se descompuso en tareas como sigue:

- Operaciones vectoriales. Líneas (5), (6) y (9) del algoritmo 3.1. Cada elemento del vector resultante es una tarea independiente.
- Productos punto. Líneas (2), (3) y (7) del algoritmo 3.1. Naturalmente esta operación correspondería a una reducción de OpenMP. Sin embargo, como se explicará en la siguiente sección, no se usaron reducciones. Las reducciones disminuyen notablemente el tiempo de ejecución pero también introducen errores de redondeo.
- Producto matriz-vector. Línea (1) del algoritmo 3.1. Una tarea T_i corresponde a un elemento del vector resultante, el cual se calcula con el producto punto del renglón *i* y el vector del lado derecho.

Flat profile:

Each 31imple counts as 0.01 seconds.

% time	cumulative seconds	self seconds	calls	self us/call	total us/call	name
82.61	5332.44	5332.44				multAb (matriz.c:180 @ 4040ba)
7.60	5823.35	490.91				multAb (matriz.c:179 @ 404135)
3.79	6068.03	244.68				multAb (matriz.c:179 @ 4040b1)
1.23	6147.14	79.11				zeros_matriz (matriz.c:227 @ 40437c)
0.95	6208.38	61.25				fprex (fprex.c:96 @ 4066eb)
0.83	6261.97	53.59				main (simp.c:274 @ 401df5)
0.77	6311.94	49.97				fprex (fprex.c:111 @ 40678c)
0.41	6338.71	26.77				zeros_matriz (matriz.c:226 @ 4043ª8)
0.19	6351.16	12.45				main (simp.c:294 @ 401f03)
0.15	6360.95	9.78				zeros_matriz (matriz.c:226 @ 404373)
0.10	6367.69	6.75				fprex (fprex.c:110 @ 4067e9)
…						
0.00	6456.88	0.00	337078	0.00	0.00	multAb (matriz.c:174 @ 40406d)
0.00	6456.88	0.00	12506	0.00	0.00	vector (matriz.c:18 @ 4034b4)
0.00	6456.88	0.00	12501	0.00	0.00	zeros_vector (matriz.c:242 @ 404447)
0.00	6456.88	0.00	10005	0.00	0.00	free_vector (matriz.c:187 @ 404158)
0.00	6456.88	0.00	10001	0.00	0.00	zeros_ivector (matriz.c:249 @ 404497)
0.00	6456.88	0.00	7501	0.00	0.00	zeros_matriz (matriz.c:222 @ 404349)

Figura 4.1. Salida del perfilador gprof.

4.4 Paralelización del método MPS usando OpenMP

La paralelización del método MPS consistió básicamente en agregar directivas de OpenMP en el código secuencial. Se agregaron las directivas de división del trabajo a los ciclos que no contienen dependencia de datos. En los casos donde existían dependencias se reescribió el código de manera que fuera posible dividir el trabajo en *chunks* o pedazos para cada hilo. Sin embargo, no fue posible incluir el ciclo global de los 2500 pasos de la función *main* puesto que tanto la posición como la velocidad de cada partícula dependen de su estado anterior y de las de sus vecinos, esto incremento la sobrecarga al entrar y salir de las regiones paralelas. El bloque de código que requirió un enfoque para ser paralelizado es el referente al producto punto que está presente en el método del gradiente conjugado y que se describe a continuación.

4.4.1 El producto punto

En el algoritmo 3.1 aparecen tres operaciones vectoriales del producto punto, líneas (2), (3) y (7). El producto punto de dos vectores $\vec{x} = \{x_1, x_2, \cdots x_n\}$ y $\vec{y} = \{y_1, y_2, \cdots y_n\}$ se define como:

$$\vec{x} \cdot \vec{y} = \sum_{i=1}^{n} x_i y_i = x_1 y_1 + x_2 y_2 + \cdots + x_n y_n$$

Cada multiplicación de las entradas correspondientes a cada vector es independiente del resto, de aquí que estas operaciones se pueden efectuar en paralelo. Esto significa que cada hilo puede calcular la suma local de los productos de los elementos correspondientes de dos vectores. Sin embargo, el resultado requiere agregar todas esas sumas locales. Ahora bien, ya que las sumas se agregan al resultado global, ellas no son independientes unas de otras. Entonces, lo que se tiene es el llamado *patrón de reducción*, en donde la operación de conmutatividad, en este caso la adición, es usada para combinar los resultados locales una vez que fueron calculados en forma paralela en la primera parte del cálculo. Si se usa la operación *reduce (operado, variable)* de OpenMP, implica que cada hilo paralelo puede agregar su resultado a la suma global con la garantía de que ningún otro hilo actualizará la suma global al mismo tiempo. Sin embargo, paralelizar el producto punto supone que el orden en el cual los resultados locales se combinan cambia entre cada ejecución. Lo anterior se debe a que cada hilo puede terminar su plan de trabajo en un tiempo diferente, y entonces cualquier hilo que esté listo primero podrá agregar su resultado a la suman global por tanto la suma global cambiará; aunque lo que es peor es que el resultado serial no podrá ser reproducido.

No obstante que los tipos de datos de punto flotante fueron introducidos en los lenguajes de programación para representar números reales, es sabido que solo son una aproximación de los mismos. Las variables de tipo de punto flotante no pueden representar todos los números reales, ya que solo se dispone de una cantidad finita de precisión (usualmente 32 o 64 bits), y por lo tanto no se garantiza la aritmética exacta. De hecho, hay muchos casos especiales en donde la aritmética de punto flotante puede fallar. Es de suma importancia el hecho de que la adición de punto flotante no es asociativa, y por lo tanto si se cambia el orden en el cual se acumulan las sumas del producto punto se pueden obtener resultados diferentes. Paralelizar el producto punto causa que el orden en que se acumulen las sumas locales sea una operación no determinística.

4.4.2 Método para sustituir el algoritmo del producto punto

Existen varios métodos para efectuar la suma de números flotantes y reducir los errores de redondeo. Un método simple consiste en ordenar de manera ascendente los números que se van a sumar y entonces sumarlos usando la operación de reducción. Aunque este algoritmo aminora los problemas de redondeo, también cambia la complejidad del problema pasando de lineal a *O(n log n)*, donde *n* es la cantidad de números que se suman. El algoritmo de Kahan es el más popular ya que el tiempo de ejecución se conserva lineal y mejora los resultados de la suma, en la práctica es lento y difícil de paralelizar. Existe otro método que se basa en el algoritmo divide y vencerás por medio de recursión, y cuya complejidad es *O(log n)*, es decir, es proporcional al logaritmo del número de elementos. La suma básica usando el algoritmo divide y vencerás se expresa en lenguaje C como:

```
float sum(const float *a, size_t n)
{
    // caso base
    if (n == 0) {
        return 0;
    }
    else if (n == 1) {
        return *a;
    }
    // caso recursivo
    size_t half = n / 2;
    return sum(a, half) + sum(a + half, n - half);
}
```

Este algoritmo no tiene un ciclo *for* de manera que su paralelización requiere un enfoque diferente. Se usa la directiva *task* de OpenMP para tratar el problema como una paralelización de tareas en lugar de paralelizar datos, la versión recursiva es la siguiente:

```
float sum(const float *a, size_t n)
{
    // caso base
    if (n == 0) {
        return 0;
    }
    else if (n == 1) {
        return 1;
    }
    // caso recursivo
    size_t half = n / 2;
    float x, y;

    #pragma omp parallel
    #pragma omp single nowait
    {
        #pragma omp task shared(x)
        x = sum(a, half);
        #pragma omp task shared(y)
        y = sum(a + half, n - half);
        #pragma omp taskwait
        x += y;
    }
    return x;
}
```

El algoritmo anterior se modificó para realizar el producto punto y se agregó un caso adicional a la recursión: el caso serial. El caso serial es necesario porque al dividir las tareas recursivamente se llega a un punto cercano al fondo del árbol de recursión donde $\frac{n}{2}$ invocaciones dividen un arreglo con dos

elementos en subtareas que solo procesan un elemento cada una, obteniendo una ejecución extremadamente lenta. Entonces, dependiendo del tamaño del arreglo por procesar se decidirá si esta tarea se realiza en paralelo o en serie.

Se agregó este algoritmo al método del gradiente conjugado para realizar las operaciones de producto punto, dando como resultado una mejora en la exactitud de los valores calculados. Para menos de 50 partículas la diferencia entre los valores de presión calculados de forma serial y paralela fue del orden de 10^{-15} en los primeros 300 pasos. No obstante, el tiempo de ejecución del algoritmo fue aún más alto que el serial aunado a que en los siguientes pasos aumento la diferencia llegando a ser del orden de 10^{-2} en el paso 2500 hecho que sucedió en el total de las 1800 partículas del fluido. Por tanto, se procedió a efectuar el producto punto de forma serial. Se debe notar que es necesario hacer un análisis más detallado de los algoritmos para sumar de números flotantes, así como de sus parámetros, actividad que quedo fuera de los alcances del presente trabajo.

4.5 Resultados

Para comprobar la exactitud y precisión del método MPS paralelo se probó usando 1, 2, 4 y 8 hilos de ejecución bajo las mismas condiciones que el algoritmo serial en cuanto a las condiciones de la columna de agua y la duración de la simulación, incluso se empleó el mismo incremento de tiempo. En cuanto a la exactitud se verificó que se obtuvieran los mismos resultados que el algoritmo serial, y en lo que corresponde a la precisión, se probó que se obtuviera el mismo resultado independientemente del número de hilos de ejecución. Los tiempos de ejecución se muestran en la Tabla 4.1. Como era de esperarse el programa de MatLab es el que tiene la peor eficiencia ya que tomo más de 25,000 segundos de tiempo de ejecución para completar la simulación. La programación serial del método MPS redujo notablemente el tiempo de ejecución en más del 50%. Por otro lado, cuando se trata de la programación paralela usando OpenMP, se observa que el número óptimo de hilos es 4 y cuyo tiempo de ejecución (2565 segundos) representa aproximadamente el 25% del tiempo consumido por el método serial (10347 segundos). En cambio, si se usan 2 y 8 hilos el tiempo de ejecución serial se reduce en 45% y 29% respectivamente.

Tipo de programación	Número de hilos	Tiempo de ejecución (seg)	Speed Up teórico	Speed Up real
Script de MatLab	1	> 25,000	--	--
Serial lenguaje C	1	10,347	--	--
Paralela C, OpenMP	1	8,803	--	--
Paralela C, OpenMP	2	4,615	1.88	1.91
Paralela C, OpenMP	4	2,565	3.39	3.43
Paralela C, OpenMP	8	2,977	5.63	2.96

Tabla 4.1. Tiempo de ejecución del método MPS serial y paralelizado.

En la tabla 4.1 se muestran las aceleraciones (speed up) teóricas y reales que se obtienen cuando se usan 2, 4 y 8 hilos. Para el cálculo del speed up teórico se usó la ley de Amdahl considerando que el código paralelizado (función multAB) representa el 82.61% del tiempo de cálculo:

$$Sp = \frac{1}{(1-p) + \frac{p}{N}} \quad (37)$$

Donde Sp es la aceleración o speed up, p es la fracción paralelizable y N es el número de procesadores.

Los valores de speed up real se calcularon simplemente dividiendo el tiempo empleado por el algoritmo serial entre el tiempo del algoritmo paralelo usando una cantidad diferente de procesadores.

Como se observa en la figura 4.2, para esta arquitectura en particular, el valor óptimo de hilos de procesamiento es de 4 ya que se alcanza una aceleración de 3.43, después de este valor la aceleración decae. Teóricamente se pueden alcanzar valores de aceleración mayores de hasta 5.75 suponiendo un número infinito de hilos en la ecuación (37).

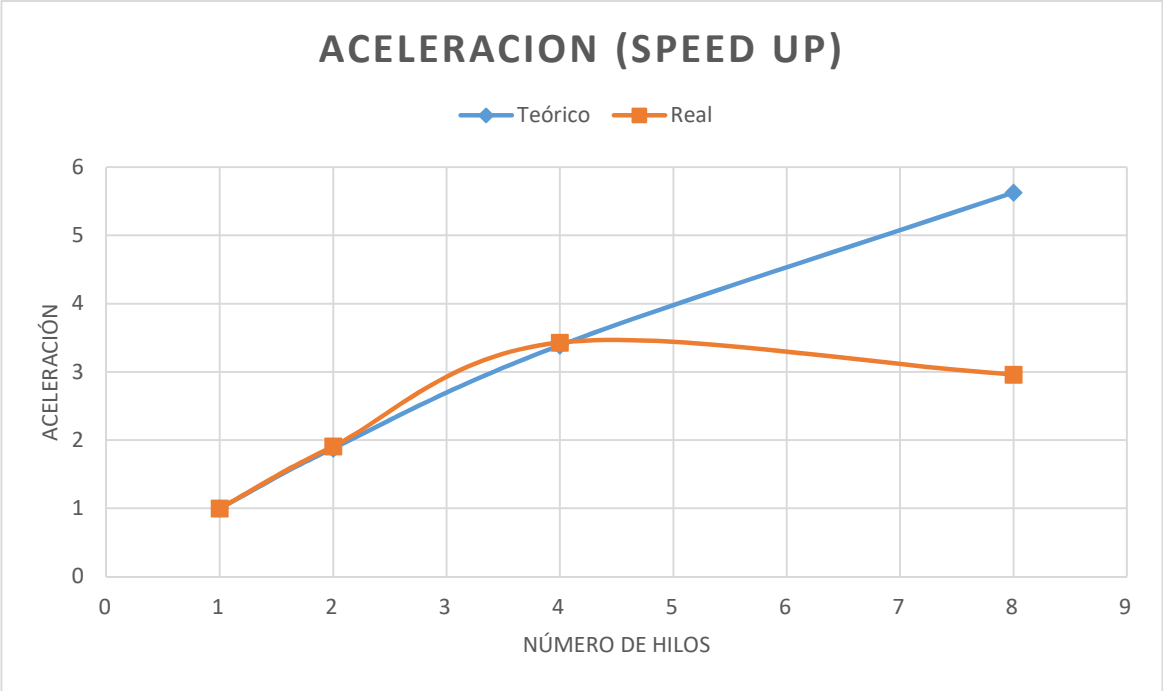


Fig. 4.2 Aceleración del código paralelo.

5 Conclusiones

Se mostraron las ecuaciones que gobiernan el movimiento de un fluido, Continuidad y Navier-Stokes. También se explicaron las fases explicita e implícita del Método de Movimiento de Partículas Semi-implícito (MPS) para obtener la posición y velocidad intermedia de cada partícula y posteriormente corregir ambas, posición y velocidad, con el gradiente de presión.

Se modelo con el método MPS el rompimiento de una columna de agua bajo la acción de la gravedad, y se llevó a cabo una simulación de 35 segundos en la que se calculó la posición y velocidad de cada partícula del fluido. A lo largo de la simulación se pudo observar la interacción entre las partículas y al

mismo tiempo calcular la presión en cada partícula; destacando la capacidad del método MPS para representar el fenómeno de chapoleo o *sloshing*.

Se programó el algoritmo MPS en forma serial y paralela para que se ejecutara en una arquitectura de memoria compartida consistente en una computadora con procesador multi-núcleo quadcore. La programación se hizo en leguaje C y en el caso paralelo se usó OpenMP. Durante las simulaciones se encontró que el tiempo que invertido por el programa serial para llevar a cabo una simulación fue de 10, 347 segundos mientras que el programa paralelo, usando 4 hilos, solo requirió de 2,565 segundos; esto es, casi una cuarta parte del tiempo serial.

Del tiempo total empleado, se encontró que la solución del sistema de ecuaciones mediante el método del gradiente conjugado representa un porcentaje alto del tiempo de ejecución, en particular las operaciones matriciales usan más del 80% del tiempo total consumido por la simulación, y por lo tanto cuando se paralelizo esta porción de código se redujo considerablemente el tiempo total de ejecución.

Finalmente, se encontró que usar la operación reducción para sumar números de punto flotante o calcular el producto punto de dos vectores provoca errores de redondeo porque la operación suma no es asociativa. El problema de la suma de números flotantes aun es tema de estudio mismo que deberá ser resuelto para las computadoras exaescala que efectúan operaciones de exaflops.

6 Referencias.

[1] R. Chandra, L. Dagun, D. Kohr, D. Maydan, J. McDonald y R. Menon. (2001). *Parallel Programming in OpenMP*. Morgan K. Pub., San Francisco, CA.

[2] Georg Hager y Gerhard Wellein. (2011). *Introduction to High Performance Computing for Scientists and Engineers*. Chapman & Hall/CRC computational science series, FL.

[3] Gingold, R.A., Monaghan, J.J. (1977). *Smoothed particle hydrodynamics: Theory and application to non-spherical stars*. Mon. Not. R. Astron. Soc. 181, 375–389.

[4] Monaghan, J.J. (1994). *Simulating free surface flows with SPH.* J. Comput. Phys. 110, 399–406.

[3] J. Sanchez-Mondragon. (2016). *On the stabilization of unphysical pressure oscillations in MPS method simulations*. Int. J. Numer. Meth. Fluids 2016; 82:471–492.

[4] Lee BH, Park JC, Kim MH, Jung SJ, Ryu MC, Kim YS. (2010). *Numerical simulation of impact loads using a particle method*. Ocean Engineering; 37:164–173.

[5] Lee BH, Park JC, Kim MH, Hwang SC. (2011). *Step-by-step improvement of MPS method in simulating violent free-surface motions and impact-loads*. Computer Methods in Applied Mechanics and Engineering; 200:1113–1125.

[6] Koshizuka S, Oka Y. (1996). *Moving-particle semi-implicit method for fragmentation of incompressible fluid*. Nuclear Science and Engineering; 123:421–434.

[7] Koshizuka S, Nobe A, Oka Y. (1998). *Numerical analysis of breaking waves using the moving particle semi-implicit method*. International Journal for Numerical Methods in Fluids 1998; 26:751–769.

[8] Vorobyev Alexander. (2013). *A Smoothed Particle Hydrodynamics Method for the Simulation of Centralized Sloshing Experiments*. Karlsruher Institut für Technologie (KIT) KIT Scientific Publishing.

[9] Löf Henrik. (2004) *Parallelizing the Method of Conjugate Gradients for Shared Memory Architectures*. IT Licentiate theses. UPPSALA UNIVERSITY Department of Information Technology.

[10] Sánchez-Mondragón Joel y Vázquez-Hernández Alberto Omar. (2018). *Solitary wave collisions by double-dam-broken simulations with the MPS method*. Engineering Computations, Engineering Computations, Vol. 35 Issue: 1, pp.53-70.

[11] Anderson Alyssa. (2014). *Achieving Numerical Reproducibility in the Parallelized Floating Point Dot Product*. Honors Theses. Paper 30.

[12] Wilkinson Barry, Allen Michael. (2005). *Parallel Programming Techniques And Applications Using Networked Workstations And Parallel Computers*. Pearson Prentice Hall.

[13] Culler, D., Singh, J., y Gupta, A. (1998). *Parallel Computer Architecture: A Hardware/Software Approach*. Morgan Kaufman.

[14] Koshizuka S., Shibata K., Kondo M. y Matsunaga T. (2018). *Moving Particle Semi-implicit Method*. *A Meshfree Particle Method for Fluid Dynamics*. Academic Press.

[15] N.J. Higham. (2002). *Accuracy and Stability of Numerical Algorithms* (2 ed). SIAM. pp. 110–123.

[16] Monaghan J.J., Lattanzio J.C. *A refined particle method for astrophysical problems*. Astronomy and Astrophysics. 1985. Vol. 149(1), p. 135–143.