Implementación (CPU-GPU) del modelo SPH para la protección de costas

J.M. DOMÍNGUEZ*, A.J.C. CRESPO, A. BARREIRO y M. GÓMEZ-GESTEIRA

*jmdominguez@uvigo.es
EPhysLab, Facultade de Ciencias,
Universidade de Vigo
Campus As Lagoas s/n, Ourense, Spain

ABSTRACT

En este trabajo se presenta la implementación del método SPH para dinámica de fluidos en GPU (unidad de procesamiento gráfico). Se analizan cuales son las mejores aproximaciones para paralelizar este modelo y traducirlo al lenguaje de programación de las GPU. Los resultados muestran grandes mejoras en el rendimiento del código cuando este se ejecuta sobre una GPU en lugar de una CPU.

Keywords: SPH, aproximación lagrangian, protección de costas, paralelización, GPU

1 Introducción

La dinámica de fluidos computacional (CFD) proporciona una valiosa herramienta para la investigación científica. Por un lado permite simular experimentos numéricos en lugar de reproducir experiencias reales que son caras y en algunas ocasiones imposibles de llevarse a cabo con modelos a escala. Por otro lado, proporciona información adicional que no se puede obtener de la observación experimental, lo cual es especialmente valioso cuando el objeto de estudio no es sólo descubrir alguna variable del flujo (como velocidad, presión, etc.) sino también entender los procesos físicos envueltos en el fenómeno.

Históricamente, la dinámica de fluidos computacional se ha centrado en métodos que se basan en una malla. Existen dos aproximaciones para describir las ecuaciones físicas que gobiernan estos métodos. La Euleriana y la Lagrangiana. Los métodos de elementos finitos (FEM) son el paradigma de los métodos Lagrangianos en los cuales se asocia una malla al material y ésta se puede deformar imitando la deformación del material. Un claro ejemplo de la descripción Euleriana son los métodos de diferencias finitas (FDM) y el método de volúmenes finitos (FVM). Éste último es el más extendido a la hora de estudiar problemas de dinámica computacional de fluidos. A pesar del éxito de estos métodos en las últimas décadas, ambas, aproximación Euleriana y Lagrangiana, presentan importantes limitaciones, incluso cuando se usan de manera conjunta. Además, el uso de métodos lagrangianos ha experimentado un gran crecimiento en la última década. En estos métodos se substituye la malla por un conjunto de nodos distribuidos de manera arbitraria. Se espera así que el modelo sea más maleable y versátil que los convencionales métodos basados en malla, especialmente para aplicaciones con grandes discontinuidades en el flujo del fluido.

El método SPH (Smoothed Particle Hydrodynamics) se desarrolló por primera vez alrededor del año 1977 para estudiar problemas astrofísicos (Lucy 1977; Gingold y Monaghan 1977) y es uno de los más conocidos métodos libres de malla. Conceptualmente, el método usa la teoría de interpolación para transformar las ecuaciones diferenciales parciales en sumatorios que describen las integrales. A pesar de su pronta aparición, el modelo no ha despertado la curiosidad de los investigadores a la hora de aplicarlo en otros campos distintos a la astrofísica hasta principios de los 90. Fue entonces cuando el modelo se aplicó con éxito en campos como el impacto en sólidos. En el caso particular de la dinámica de fluidos, quizás el mayor logro de la técnica SPH ha sido su aplicación a problemas de fluidos con superficie libre.

El grupo denominado SPHysics (www.sphysics.org) ha centrado su investigación en la propagación de las olas y en su interacción con estructuras costeras, tanto en 2D (Gómez-Gesteira et al. 2005; Dalrymple y Rogers 2006; Crespo et al. 2008a) como en 3D (Gomez-Gesteira y Dalrymple 2004; Crespo et al. 2007a).

En este trabajo se prestará especial atención en las capacidades del método para resolver problemas con superficie libre en general y problemas con olas en particular. Se analizarán diferentes mejoras y correcciones sobre el modelo clásico y el modelo SPHysics (Gómez-Gesteira et al. 2010) se usará como ejemplo de técnica SPH mejorada para analizar diferentes aplicaciones.

2 Smoothed Particle Hydrodynamics: Teoría

En la técnica SPH, el dominio del fluido se representa por un conjunto de puntos o nodos repartidos en el espacio en donde se conocen diferentes propiedades físicas (masa, densidad, velocidad, posición, presión). Estos puntos se mueven describiendo el fluido de manera Lagrangiana y sin la necesidad de describir una malla. Las propiedades pueden cambiar con el tiempo debido a interacciones entre puntos o partículas que estén más próximas.

2.1 Método de interpolandos

El método de interpolación en SPH se basa en la siguiente integral:

$$f(\mathbf{s},t) = \int_v W(\mathbf{s}-\mathbf{x},h)\, f(\mathbf{x},t)\, dv \quad (1)$$

donde la integral es sobre el dominio *v*, *dv* es el elemento de volumen que depende de las dimensiones del problema y $W(\mathbf{s}-\mathbf{x},h)$ es la función llamada *kernel* en SPH. El parámetro *h* determina el tamaño del kernel, es decir, el radio de influencia alrededor de *s*. En 3D, este dominio se corresponde con una esfera de radio *nh*, donde *n* depende de la definición del kernel. Aunque *h* es una constante en las aplicaciones más simples de SPH, este parámetro también puede variar en el tiempo y en el espacio.

Las integrales se sustituyen numéricamente por sumatorios de las contribuciones de las partículas cercanas en ese dominio:

$$f(\mathbf{s},t) \approx \sum_j W(\mathbf{s}-\mathbf{x}_j,h)\, f_j\, \Delta v_j \quad (2)$$

tal que

$$\sum_j W(\mathbf{s}-\mathbf{x}_j,h)\, \Delta v_j = 1 \quad (3)$$

donde Δv_j es el volumen asociado a la partícula *j*. En SPH la masa, m_j, de la partícula *j*, permanece fija, mientras que la densidad, ρ_j, puede variar.

En la formulación clásica de SPH, cuando se resuelven las ecuaciones de Navier-Stokes, se considera que el fluido es compresible.

En general, se remplaza el volumen por $\Delta v_j = m_j/\rho_j$. De este modo la ecuación (2) quedaría:

$$f(\mathbf{s},t) \approx \sum_j \frac{m_j}{\rho_j} W(\mathbf{s}-\mathbf{x}_j,h)\, f_j \quad (4)$$

2.2 Kernels

Además, la función *kernel* tiene que cumplir una serie de propiedades:

i) Positividad:

$W(\mathbf{s}-\mathbf{x},h) \rangle 0$ dentro del dominio Ω

ii) Soporte compacto:

$W(\mathbf{s}-\mathbf{x},h) = 0$ fuera del dominio Ω

iii) Normalización:

$$\int_v W(\mathbf{s}-\mathbf{x},h)\, dv = 1$$

iv) Comportamiento tipo función delta:

$$\lim_{h\to 0} W(\mathbf{s}-\mathbf{x},h)\, dv = \delta(\mathbf{s}-\mathbf{x})$$

v) Comportamiento monótono decreciente de $W(\mathbf{s}-\mathbf{x},h)$

Hay una gran variedad de posibles funciones kernel. El kernel Gaussiano ha sido considerado por diversos autores aunque es de clase C^{-1}. Ya que no se vuelve cero a ninguna distancia finita. Se puede usar un kernel Gaussiano renormalizado con limite de corte (*3h*) para asegurar la tercera propiedad. Algunos autores también usan el kernel cuadrático. El comportamiento de su primera derivada indica que las fuerzas de interacción entre partículas aumentan a medida que la distancia entre ellas se reduce. Se ha demostrado que para el estudio de propagación de olas no lineales, usar un kernel cuadrático conserva las propiedades de las olas en la superficie libre. Posiblemente, el kernel más conocido y usado sea el cubic-spline. En general, la precisión de la

interpolación SPH aumenta con el orden de los polinomios usados para definir el kernel, sin embargo, el tiempo de computación también aumenta. El kernel quíntico (Fig. 1) y el kernel llamado Wendland constituyen una buena opción ya que proporciona un orden de interpolación mayor con un coste computacional comparable al del kernel cuadrático.

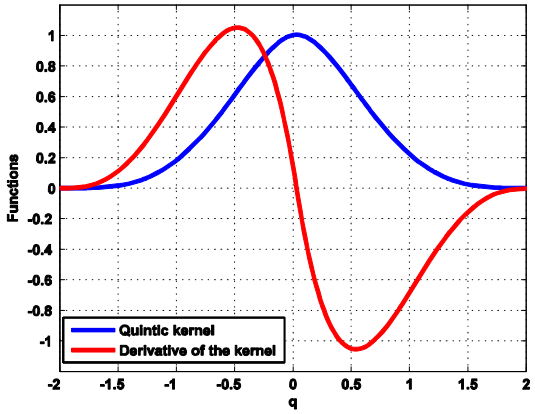


FIG. 1. Kernel quíntico y su derivada.

2.3 Derivadas

Una de las ventajas más importantes del SPH es que el kernel es una función diferenciable. Las derivadas de esta interpolación se pueden obtener por diferenciación ordinaria. (no hay necesidad de usar diferencias finitas) y tampoco es necesario el uso de una malla. Entonces, si *W* es una función diferenciable y queremos aplicar el operador ∇ a la función A_i, simplemente tendremos que aplicar el operador ∇ a la función kernel.

Así el gradiente de la función escalar *A* se puede calcular como

$$\nabla A_i(\mathbf{s},t) = \sum_j \frac{m_j}{\rho_j} \nabla W(\mathbf{s}-\mathbf{x}_j, h)\, A_j \qquad (5)$$

Según la segunda regla de oro del SPH

$$\rho \nabla A = \nabla(\rho A) - A\nabla\rho \qquad (6)$$

el gradiente de *A* en la partícula *i* se puede calcular como

$$\nabla A_i(\mathbf{s},t) = \sum_j \frac{m_j}{\rho_j} (A_j - A_i) \nabla W(\mathbf{s}-\mathbf{x}_j, h) \qquad (7)$$

Se puede usar una expresión similar para calcular la divergencia de cualquier función vectorial (por ejemplo velocidad *u*).

$$\mathrm{div}(\mathbf{u}_i) = \sum_j \frac{m_j}{\rho_j} (\mathbf{u}_j - \mathbf{u}_i) \nabla W(\mathbf{s}-\mathbf{r}_j, h) \qquad (8)$$

3 Ecuaciones de conservación y estado

En general, el fluido se considera ligeramente compresible en SPH, aunque algunos autores también han desarrollado una versión incompresible del modelo. A continuación se describirá la versión compresible, que es la aproximación más utilizada para modelar flujos de agua en SPH.

3.1 Conservación de la masa

La ley de conservación para un fluido se puede escribir en la forma Euleriana como

$$\frac{\partial \rho}{\partial t} + \mathrm{div}(\rho \mathbf{u}) = 0 \qquad (9)$$

En la forma Lagrangiana, la masa del fluido asociada al nodo *i* es constante. Así la expresión anterior se puede reescribir de la forma

$$\frac{1}{\rho}\frac{d\rho}{dt} = -\mathrm{div}(\mathbf{u}) \qquad (10)$$

la cual se puede integrar sobre el dominio *v* después de multiplicar ambos lados de la ecuación por el kernel

$$\int_v \frac{1}{\rho}\frac{d\rho}{dt} W(\mathbf{s}-\mathbf{x}, h)\, dv = -\int_v \mathrm{div}(\mathbf{u})\, W(\mathbf{s}-\mathbf{x}, h)\, dv \qquad (11)$$

Por la propia definición de kernel, el término de la izquierda es $(1/\rho)(d\rho/dt)$ evaluado en la posición *s* y el término de la derecha se puede reescribir como

$$-\int_v [\mathrm{div}(W(\mathbf{s}-\mathbf{x},h)\,\mathbf{u}) - \mathbf{u}\,\nabla W(\mathbf{s}-\mathbf{x},h)]\,dv \quad (12)$$

Usando el teorema de Gauss

$$\frac{1}{\rho}\frac{d\rho}{dt} = \int_v \mathbf{u}\,\nabla W(\mathbf{s}-\mathbf{x},h)\,dv - \int_S W(\mathbf{s}-\mathbf{x},h)\,\mathbf{u}\,\mathbf{n}\,dS \quad (13)$$

donde la última integral es una integral de superficie sobre la superficie *S* que encierra el volumen *v* y *n* un vector unitario normal a la superficie y apuntado hacia afuera. Esta integral se puede despreciar cuando la superficie está alejada del punto ***s*** ya que la función kernel decrece a cero rápidamente.

En la forma continua la ecuación queda

$$\frac{1}{\rho}\frac{d\rho}{dt} = \int_v \mathbf{u}\,\nabla W(\mathbf{s}-\mathbf{x},h)\,dv \quad (14)$$

la cual se reescribirá en forma discreta como

$$\left(\frac{1}{\rho}\frac{d\rho}{dt}\right)_i = \sum_j \frac{m_j}{\rho_j}\mathbf{u}_j \nabla_j W_{ij} \quad (15)$$

Debido a la simetría del gradiente, su primera derivada cumple lo siguiente $\nabla_i W_{ij} = -\nabla_j W_{ij}$

Con lo que esto resulta en

$$\left(\frac{1}{\rho}\frac{d\rho}{dt}\right)_i = -\sum_j \frac{m_j}{\rho_j}\mathbf{u}_j \nabla_i W_{ij} \quad (16)$$

Una vez más, la ec. (6) se puede aplicar para obtener una ecuación donde la evolución de la variable bajo estudio dependa de la diferencia entre las partículas *i* y *j*. El gradiente de una constante es cero, así

$$0 = \sum_j \frac{m_j}{\rho_j}\nabla_i W_{ij} \quad (17)$$

Multiplicando por u_i, este término puede ser añadido a la ec. (16) obteniendo

$$\left(\frac{d\rho}{dt}\right)_i = \rho_i \sum_j \frac{m_j}{\rho_j}(\mathbf{u}_i - \mathbf{u}_j)\nabla_i W_{ij} \quad (18)$$

Una aproximación diferente sería

$$\rho_i = \sum_j m_j \nabla_i W_{ij} \quad (19)$$

la cual permite calcular la densidad como una suma sobre las partículas vecinas en lugar de resolver una nueva ecuación diferencial. Aún así, esta última ecuación no es la mejor opción para el cálculo de fluidos ya que lleva a una caída del valor de la densidad cerca de la superficie del fluido. Por otro lado, usando la ec. (18) la densidad varía cuando las partículas se mueven las unas respecto a las otras.

3.2 Conservación del momento

La ecuación que gobierna el movimiento del fluido se puede derivar de manera discreta usando el mismo protocolo que para la conservación de la masa. Si consideramos el sistema en ausencia de viscosidad, la ecuación de movimiento en forma Lagrangiana es

$$\frac{d\mathbf{u}}{dt} = -\frac{1}{\rho}\nabla p - \mathbf{g} \quad (20)$$

Se puede considerar el siguiente cambio de variable para simetrizar el gradiente de presión

$$\frac{\nabla p}{\rho} = \nabla\left(\frac{p}{\rho}\right) + \frac{p}{\rho^2}\nabla\rho \quad (21)$$

De este modo la ecuación del momento para la partícula *i* queda

$$\left(\frac{d\mathbf{u}}{dt}\right)_i = -\sum_j m_j\left(\frac{p_j}{\rho_j^2} + \frac{p_i}{\rho_i^2}\right)\nabla_i W_{ij} - \mathbf{g} \quad (22)$$

Esta expresión es, posiblemente, la más usada para describir el gradiente de presiones en SPH aunque otros autores consideren que la expresión

$$-\sum_j \frac{1}{\rho_i \rho_j} m_j (p_j + p_i)\nabla_i W_{ij} \quad (23)$$

para el gradiente de presión es variacionalmente consistente con el uso de la ec. (10).

3.3 Viscosidad

La viscosidad juega un papel clave a la hora de prevenir inestabilidades en el movimiento del fluido, donde partículas aisladas pueden llegar a moverse de

manera caótica. Estas inestabilidades se pueden evitar usando términos viscosos, los cuales se deben añadir a la ecuación del momento descrita anteriormente. Se han considerado diferentes tratamientos para describir la viscosidad en la literatura, pero aquí nos centraremos en los más extendidos.

3.3.1 Viscosidad artificial

La Viscosidad artificial se ha usado muy a menudo debido a su simplicidad. En notación SPH, la ecuación de momento se puede reescribir como

$$\frac{d\mathbf{u}_i}{dt} = -\sum_j m_j \left(\frac{p_j}{\rho_j^2} + \frac{p_i}{\rho_i^2} + \Pi_{ij} \right) \nabla_i W_{ij} - \mathbf{g} \quad (24)$$

donde Π_{ij} es el término viscoso:

$$\Pi_{ij} = \begin{cases} \dfrac{-\alpha \overline{c}_{ij} \mu_{ij}}{\overline{\rho}_{ij}} & \mathbf{u}_{ij} \mathbf{x}_{ij} < 0 \\ 0 & \mathbf{u}_{ij} \mathbf{x}_{ij} > 0 \end{cases}$$

con $\mu_{ij} = (h\mathbf{u}_{ij} \mathbf{x}_{ij}) / (r_{ij}^2 + \eta^2)$, $\mathbf{u}_{ij} = \mathbf{u}_i - \mathbf{u}_j$, $\mathbf{x}_{ij} = \mathbf{x}_i - \mathbf{x}_j$ y $\overline{c}_{ij} = (c_j + c_i)/2$ es la velocidad media del sonido. Se incluye el parámetro $\eta^2 = 0.01h^2$ para evitar singularidades y α es un parámetro libre que se puede cambiar acorde al problema de estudio.

3.3.2 Viscosidad laminar

Otra formulación para la viscosidad es aquella que trata problemas con fluidos en los que se encuentran números de Reynolds bajos. La ecuación del momento se reescribe en este caso como

$$\frac{d\mathbf{u}_i}{dt} = -\sum_j m_j \left(\frac{p_j}{\rho_j^2} + \frac{p_i}{\rho_i^2} \right) \nabla_i W_{ij} - \mathbf{g} + \sum_j m_j \left(\frac{4\upsilon_0 \mathbf{x}_{ij} \nabla_i W_{ij}}{(\rho_i + \rho_j)(r_{ij}^2 + \eta^2)} \right) \mathbf{u}_{ij} \quad (25)$$

donde, en general, la viscosidad cinemática del fluido laminar υ_0 es del orden de $10^{-6}\ m^2 s^{-1}$ para el caso del agua.

3.3.3 Viscosidad laminar y turbulencia de escala sub-partícula (SPS)

La viscosidad artificial representa la viscosidad del fluido y evita la interpenetración de partículas. Además ayuda a mantener la estabilidad numérica del esquema computacional. Sin embargo, el método es demasiado dispersivo afectando a la tensión y a la propagación del fluido en algunas ocasiones, especialmente cuando éste no es dominado por la componente gravitacional.

Por otro lado, la viscosidad laminar es precisa para números de Reynolds bajos, pero no capta las características importantes del fluido cuando se involucran los términos turbulentos.

Usando los conceptos de Large Eddy Simulation (LES), (Dalrymple y Rogers 2006) aplicaron la aproximación Sub-Particle Scale (SPS) para modelar la turbulencia y así representar los efectos de la turbulencia en su SPH compresible. SPS para fluidos compresibles requiere métodos de medias espaciales como Favre-averaging. La ecuación de conservación del momento se puede escribir como

$$\frac{d\mathbf{u}}{dt} = -\frac{1}{\rho}\nabla p - \mathbf{g} + \upsilon_0 \nabla^2 \mathbf{u} + \frac{1}{\rho}\nabla \tilde{\tau} \quad (26)$$

donde los tres primeros términos de la derecha (gradiente de presión, gravedad y términos laminares) pueden tratarse siguiendo la ec. (25) y $\tilde{\tau}$ representa el tensor SPS dado por

$$\tilde{\tau}_{\mu\nu} = \overline{\rho}\left(2\upsilon_t \tilde{S}_{\mu\nu} - \frac{2}{3}\tilde{S}_{\kappa\kappa}\delta_{\mu\nu} - \frac{2}{3} C_I \Delta l^2 \delta_{\mu\nu} \right) \quad (27)$$

donde $C_I = 0.0066$, (Dalrymple y Rogers 2006), Δl es el espaciado entre partículas y

$$\tilde{S}_{\mu\nu} = -\frac{1}{2}\left(\frac{\partial u_\mu}{\partial x_\nu} + \frac{\partial u_\nu}{\partial x_\mu} \right) \quad (28)$$

Se usa un modelo Smagorinsky para determinar la viscosidad Eddy

$$\upsilon_t = [\min(C_s \Delta l)]^2 |\tilde{S}| \quad (29)$$

Siendo C_s=0.12 la constante de Smagorinsky y $|\tilde{S}| = (2S_{\mu\nu}S_{\mu\nu})^{1/2}$. Así, la tensión SPS se discretiza de manera simétrica como

$$\frac{1}{\rho_i}\left(\frac{\partial \tilde{\tau}_{\mu\nu}}{\partial x_\nu}\right)_i = \sum_j m_j \left(\frac{(\tilde{\tau}_{\mu\nu})_j}{\rho_j^2} + \frac{(\tilde{\tau}_{\mu\nu})_i}{\rho_i^2}\right)\frac{\partial W_{ij}}{\partial x_\nu} \quad (30)$$

3.4 Ecuación de estado

La ecuación de estado se utiliza para calcular la relación entre la densidad y la presión del fluido, y tiene la siguiente forma:

$$p = B\left[\left(\frac{\rho}{\rho_0}\right)^\gamma - 1\right] \quad (31)$$

donde γ = 7 y $B = \frac{c_0^2 \rho_0}{\gamma}$, siendo ρ_0 = 1000 kg m^{-3} la densidad de referencia y $c_0 = c(\rho_0)$ la velocidad del sonido en la densidad de referencia.
El uso de valores de c_0 altos es adecuado para un fluido débilmente compresible. Por razones numéricas, el valor de c_0 debe ser lo suficientemente alto para reducir las fluctuaciones relativas de la densidad, las cuales son proporcionales al número de Mach al cuadrado.
El SPH estándar se usa también para resolver flujos compresibles. El movimiento de las partículas es conducido por la fuerza del gradiente de presión, calculando la presión de las partículas a partir de la densidad local de cada una de ellas, y por la energía interna a través de la ecuación de estado.

3.5 Corrección XSPH

Las partículas se mueven usando

$$\frac{d\mathbf{r}_a}{dt} = \mathbf{v}_a \quad (32)$$

o el variante XSPH (donde "X" es un factor desconocido).

$$\frac{d\mathbf{r}_a}{dt} = \mathbf{v}_a + \varepsilon \sum_b m_b \left(\frac{\mathbf{v}_{ab}}{\bar{\rho}_{ab}}\right) W_{ab} \quad (33)$$

donde, $\bar{\rho}_{ab} = \frac{\rho_a + \rho_b}{2}$ y, ε es una constante que toma valores entre cero y uno (normalmente el valor que se toma es el de 0.5).
El XSPH es una corrección para la velocidad de una partícula. Así a la velocidad de la partícula se le suma un promedio de las velocidades de todas las partículas que actúan sobre ella. Debido al rango finito del kernel, el sumatorio sólo afecta a los vecinos más próximos.
Esta corrección de la velocidad mantiene las partículas más ordenadas, y para velocidades de flujo altas se evita la penetración de un fluido en otro (en caso de que se tuviesen varios).

4 Implementación

4.1 Condiciones de contorno

Las condiciones de contorno no aparecen de forma natural en el formalismo SPH. Cuando una partícula se aproxima a una frontera sólida, en los sumatorios sólo intervienen las partículas situadas en el interior del medio, sin ningún tipo de interacción proveniente del exterior. Esta contribución puede generar efectos no realistas, debido a la diferente naturaleza de las variables a resolver, ya que algunas, como la velocidad, decaen a cero al acercarse al contorno, mientras que otras, como la densidad, no. Las diferentes soluciones para evitar problemas de contorno consisten en la creación de una serie de partículas virtuales que caractericen los límites del sistema. Pueden distinguirse básicamente tres tipos de partículas contorno, dependiendo de los autores:
Partículas fantasmas. Se consideran unas partículas contorno cuyas propiedades, incluida su posición, varían en cada paso temporal. Si en un cierto instante t_i la partícula *a* se encuentra a

una distancia menor que la distancia de suavizado del kernel de un contorno, entonces se genera una partícula virtual fuera del medio, constituyendo la imagen especular de la partícula incidente. Ambas partículas tienen la misma densidad y presión, pero velocidad opuesta. Este tipo de condiciones de contorno no evita totalmente el paso de partículas fluidas a través de las paredes, además, el número de partículas contorno varía en cada paso de cálculo, lo que complica la implementación de los bucles. Una de las mayores desventajas es que el número de partículas contorno varía a lo largo de la simulación lo cual complica la implementación del código. Además el modelado de geometrías muy complejas es mucho más costoso con esta técnica.

Partículas repulsivas. Un método usual para resolver los contornos en las simulaciones es descomponer las paredes en partículas (partículas contorno) las cuales ejercerán fuerzas centrales repulsivas sobre las partículas fluidas. Así, para una partícula contorno y una partícula fluida separadas una distancia *r* la fuerza por unidad de masa tiene la forma del potencial de Lennard-Jones

De forma análoga, otros autores expresan esta fuerza asumiendo la existencia de fuerzas intensas en los contornos, las cuales pueden ser descritas por una función delta.

Partículas dinámicas. Estas partículas siguen las mismas ecuaciones de continuidad y de estado que las fluidas, pero su posición permanece constante o es impuesta externamente. Una ventaja interesante de este tipo de partículas es su simplicidad computacional ya que podrán ser calculadas dentro de los mismos bucles que se usan para las partículas fluidas.

El método está descrito en detalle en (Crespo et al. 2007b) y (Gómez-Gesteira et al. 2010).

En resumen, la implementación de las condiciones de contorno es un frente abierto en SPH y se deben conducir nuevas investigaciones por este camino.

4.2 Paso de tiempo variable

Para resolver las ecuaciones a lo largo del tiempo, los esquemas SPH usan métodos de integración, los cuales deben de ser al menos de segundo orden. Los más usados son los métodos predictor-corrector y el método de Verlet. El paso de cálculo es muy dependiente de las propiedades del fluido. Así, por ejemplo, el paso de cálculo desciende cuando el fluido colisiona con contornos fijos, ya que las fuerzas aumentan de repente o durante la rotura de una ola donde los valores de velocidad también aumentan. En general, el paso de tiempo depende de la condición CFL, de los términos de fuerza y del término de difusión viscosa. Se calcula un paso de tiempo variable Δt acorde a:

$$\Delta t = C \cdot \min(\Delta t_f, \Delta t_{cv}) \tag{34}$$

$$\Delta t_f = \min_i \left(\sqrt{h / |f_i|} \right) \tag{35}$$

$$\Delta t_{cv} = \min_i \frac{h}{c_s + \max_j \left| \frac{h \mathbf{v}_{ij} \mathbf{x}_{ij}}{r_{ij}^2} \right|} \tag{36}$$

Aquí Δt_f se basa en la fuerza por unidad de masa $|f_i|$, y Δt_{cv} controla la condición de Courant y la viscosidad del sistema. C es una constante del orden de 0.1.

4.3 Eficiencia computacional: Lista de vecinos

Cada partícula de fluido necesita una lista de vecinos dentro de una distancia, que será el rango del Kernel (*2h* para el caso del cubic-spline). La lista entera, que se actualiza en cada paso de tiempo, requiere un número de operaciones del orden de N^2 para calcular las interacciones entre todas las parejas de

partículas, donde N es el número de partículas.
En el código se divide el domino computacional en celdas cuadradas de lado *2h*. Así, para una partícula localizada dentro de una celda dada, sólo será necesario considerar las interacciones con las partículas de celdas vecinas. De este modo el número de cálculos y, por lo tanto, el tiempo del mismo disminuyen considerablemente, pasando de N^2 operaciones a N·logN. Esto resulta en un considerable ahorro de tiempo de ejecución.

4.4 Implementación del modelo en CPU

Todas las ecuaciones y aproximaciones descritas en el apartado anterior están implementadas en SPHysics. El código está organizado en tres pasos fundamentales: creación de lista de vecinos (LV), cómputo de fuerzas de interacción entre partículas (CF) y cálculo de las variables de cada partícula en el siguiente paso de la simulación (CP). Por tanto, la ejecución de una simulación usando el código SPHysics consiste básicamente en un proceso iterativo en el que se repiten estos tres pasos hasta completar el tiempo de simulación requerido.
Primero se genera la lista de vecinos, para ello se implementa la lista Cell-linked descrita en (Dominguez et al. 2010). Este algoritmo se caracteriza por obtener un rendimiento óptimo con un consumo de memoria mínimo. En el código se divide el domino computacional en celdas cuadradas de lado *2h* (rango del kernel). Así, para una partícula localizada dentro de una celda dada, sólo será necesario considerar las interacciones con las partículas de celdas vecinas. En este procedimiento no se crea una lista de vecinos por cada partícula, sino que se genera una lista de todas las partículas ordenadas según la celda a la que pertenecen. Además se almacena en un vector la posición de la primera partícula de cada celda en dicha lista. Finalmente, se reordenan todos los datos que conforman las partículas (posición, velocidad, presión,…) consiguiendo así mejorar el patrón de acceso a la memoria del siguiente paso.
El segundo paso es el cómputo de fuerzas, en él se calcula la interacción de cada partícula con sus vecinas. Sólo las partículas que estén a una distancia menor que 2h entran en la interacción y sólo las partículas de la propia celda y de las celdas contiguas son candidatas a serlo. Para mejorar el rendimiento en este paso se explota la simetría existente en el cálculo de fuerzas. Al computar la fuerza que una partícula ejerce sobre otra, también se obtiene la fuerza de la segunda sobre la primera puesto que son iguales y de signo contrario.
Por ello sólo es necesario realizar la búsqueda de vecinas en la mitad de celdas adyacentes como se puede ver en la Fig. 2. Esta aproximación es clave porque consigue duplicar la velocidad de ejecución.

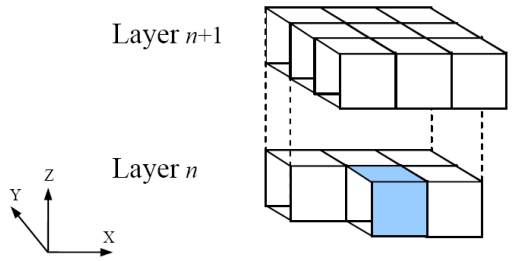


FIG. 2. Celdas empleadas en la búsqueda de vecinos. Esta figura se corresponde con la figura 2.3 de la guía de usuarios SPHysics (Gómez-Gesteira et al. 2010).

En el último paso se calcula el tiempo de simulación y con él se actualiza el estado de las partículas correspondientes al instante actual.
Para analizar el tiempo de ejecución empleado por cada uno de los pasos descritos, se ejecutó una simulación para que nos sirva de caso de referencia. El caso de estudio consiste en el colapso de un volumen de agua debido a la gravedad y su interacción con una

estructura rectangular Fig. 3. Una configuración numérica similar se usó en estudios previos para mostrar la precisión del modelo (Gomez-Gesteira & Dalrymple 2004).

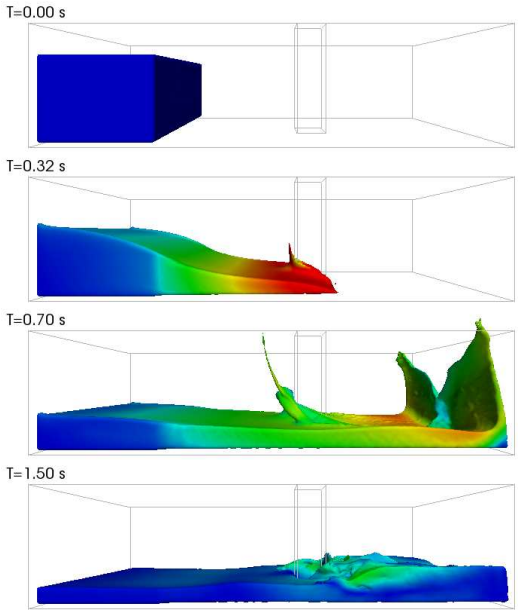


FIG. 3. Diferentes instantes de la evolución del fluido en el caso de estudio.

La Figura. 4 muestra el reparto del tiempo de ejecución entre los tres pasos en los que se divide el proceso SPH. El cómputo de fuerzas supone más del 99% del tiempo total y por tanto será el primer objetivo a la hora de buscar nuevos métodos que reduzcan el tiempo de cómputo. Al aumentar el número de partículas los porcentajes asignados a cada paso se mantienen constantes.

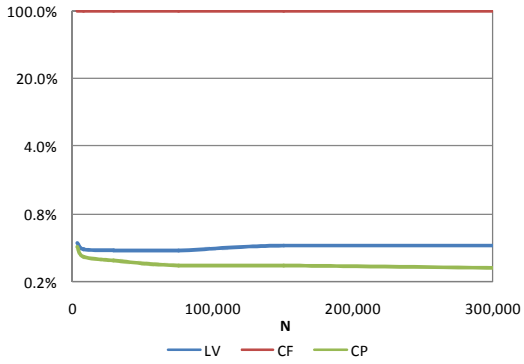


FIG. 4. Porcentaje del tiempode simulación de cada paso llevado a cabo por las diferentes partes de SPH.

4.5 Implementación del modelo en GPU

El modelo SPH posee una carga computacional muy elevada. El tiempo de ejecución crece exponencialmente con el número de partículas. Debido a ello se hace imprescindible buscar métodos de paralelización que hagan posible simulaciones con un elevado número de partículas.

Debido al empuje del mercado multimedia y de los videojuegos, las GPUs (unidades de procesamiento gráfico) han evolucionado enormemente en los últimos años y su poder de computación es mucho mayor que la CPU. Las GPUs están diseñadas para tratar operaciones sobre grandes flujos de datos y para renderizar pixeles (varias decenas de imágenes por segundo). Desarrolladores e investigadores están encontrando innumerables aplicaciones prácticas para la tecnología GPU en campos como el procesamiento de vídeo, astrofísica, mecánica de fluidos, biología, química, etc.

Por lo tanto el alto rendimiento de estos dispositivos se puede aprovechar en el ámbito científico para paralelizar los códigos existentes.

Por tanto, las GPUs constituyen un hardware muy conveniente para cálculos de carácter científico donde se realizan operaciones matemáticas a un volumen grande de datos y donde el flujo de operaciones es relativamente sencillo. La programación de GPUs para propósito general requiere el uso de lenguajes específicos, así para las tarjetas de AMD se necesita usar Brook+ y CUDA para las de nVidia (opción elegida para este trabajo). CUDA es un lenguaje de programación en paralelo con algunas extensiones sobre C/C++. El software necesario para usar CUDA (ejemplos, compiladores, entornos gráficos, etc.) se pueden descargar gratuitamente de la página web principal de nVidia (http://www.nvidia.com/cuda). Una

mayor descripción se puede encontrar en la guía de CUDA.

En la sección anterior se concluyó que la parte del proceso SPH que más tiempo consumía era el cómputo de fuerzas. Así resulta imprescindible buscar una solución que explote el paralelismo de la GPU para este proceso. Se podría mantener los otros dos pasos (LV y CP) en CPU, pero en cada iteración de la simulación sería necesario transferir todos los datos de las partículas así como la información relativa a la lista de vecinos, calcular las fuerzas y posteriormente recuperar los resultados de la memoria de la GPU. La opción más eficiente es mantener todos los datos en la memoria del dispositivo, lo cual implica realizar todos los procesos de forma paralela dentro de la GPU y transferir a la CPU sólo los resultados que se vayan a escribir en el disco duro. La grabación de los resultados representa menos del 0.01%, este valor es tan bajo puesto que sólo se realiza una vez cada muchas iteraciones o pasos de cálculo.

La creación de la lista de vecinos (LV) usa el mismo modelo conceptual que el empleado en CPU. Este método se divide en cuatro operaciones: calcular la celda en que reside cada partícula, ordenar las partículas en función de su celda (para lo cual se usa el algoritmo radixsort disponible en los ejemplos de CUDA]), generar un vector con la posición de la primera partícula de cada celda y ordenar todos los datos de las partículas. El proceso de generar el vector con la posición de las primera partícula de las celdas es el que mayores complicaciones presenta en la programación en paralelo puesto que es una operación secuencial en la que cada valor se obtiene una vez obtenido el anterior.

La implementación en GPU del cómputo de paso (CP) no presenta complicaciones, debido a que este paso está formado por operaciones que fácilmente se pueden paralelizar como es el actualizar los datos de todas las partículas para el siguiente instante en función de los valores actuales, las fuerzas entre partículas y el paso de cálculo.

Para mejorar el rendimiento del cálculo, el cómputo de fuerzas (CF) debe ser implementado en paralelo. En (Crespo et al. 2009) se estudiaron las mejores aproximaciones para la implementación de la interacción entre partículas en GPU. Se analizó el uso de la memoria compartida para reducir el acceso a la memoria global de la GPU. Sin embargo muchas de estas técnicas no son viables una vez se implementan todos los pasos del método SPH para tener una simulación completa. Así, cuando el número de partículas de la simulación es elevado, el tamaño de la memoria caché resulta insuficiente para almacenar todas las variables de las partículas de tan siquiera una celda. La técnica finalmente empleada consiste en que cada hilo de ejecución calcula, para una sola partícula, la fuerza resultante de la interacción con todas sus vecinas. Sin embargo aparecen varios problemas. Por un lado, la carga de trabajo de los hilos dentro de un bloque no está balanceada ya que cada partícula puede tener diferente número de vecinos. Por otro lado, existe divergencia porque al evaluar los posibles vecinos, en algunos casos estos son vecinos reales y se realiza el cómputo de fuerzas y en otros no. Otro problema es el patrón de acceso a la memoria global que es irregular, no hay forma de organizar los datos para que el acceso resulte coalescente para todas las partículas. También es importante resaltar que no se puede aplicar la simetría en el cálculo de fuerzas como se hacía en el código de CPU, ya que esto implicaría que cada hilo calcule parte de la fuerza de interacción de otros hilos y la sincronización en este caso supondría

una pérdida de rendimiento muchísimo mayor.

La Figura 5 muestra al igual que la Figura 4, el reparto del tiempo de ejecución en los procesos SPH pero esta vez ejecutado en la GPU. Los resultados obtenidos son similares aunque el cómputo de fuerzas representa ahora el 96% frente al 99% obtenido en CPU.

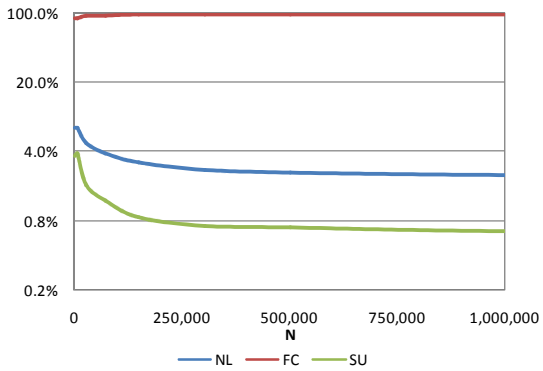


FIG. 5. Porcentaje del tiempo de simulación de cada paso llevado a cabo por las diferentes partes de SPH en GPU.

A continuación se muestran los resultados obtenidos usando la versión GPU de SPHysics. La FIG. 5 muestra una comparativa de los tiempos de computación empleados para ejecutar la misma simulación en CPU y en GPU. El caso de estudio es el descrito anteriormente. Se ejecutó el caso para distinto número de partículas en CPU y GPU simulando 1.5 segundos de tiempo físico real. En la figura se observa como la implementación con CUDA consigue reducir significativamente el tiempo de cálculo. Las simulaciones se han realizado bajo el sistema operativo Ubuntu v9.04 y en el mismo equipo, cuyas características son las siguientes: INTEL Core i7 940 a 2.93 GHz, 6 GB de RAM DDR3 a 1333 Mhz, Placa base GA-EX58-UD3R GIGABYTE, NVIDIA GeForce 8400GS y NVIDIA GeForce GTX 285 (dedicada sólo a cálculo).

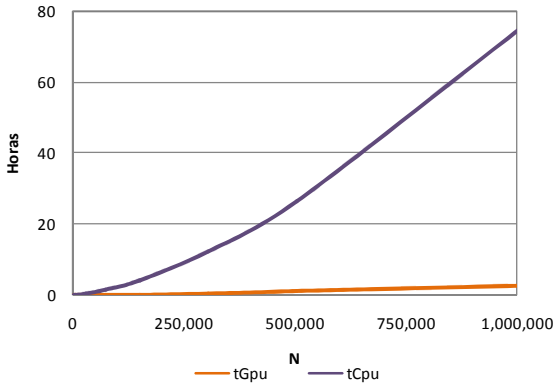


FIG. 6. Tiempo de ejecución del caso de referencia con distinto número de partículas en CPU y en GPU.

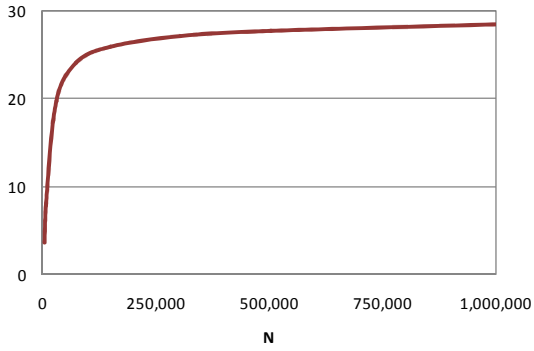


FIG. 7. Speedup conseguido del método SPH con la implementación en GPU.

A partir de 300,000 partículas el código en GPU es unas 30 veces más eficiente que el de CPU. La FIG. 6 muestra todos los speed-ups para diferente número de partículas en la simulación.

5 Caso de estudio

El paso periódico de tormentas cerca de las zonas costeras da lugar a olas peligrosas en la línea de costa. Por ejemplo, la costa española se vio afectada por grandes olas y fuertes vientos el 12 de enero de 2008. En la costa de A Coruña olas de 12 metros de altura destruyeron el dique Fira 8. Varias personas resultaron heridas y un cuantioso daño fue causado en el mobiliario urbano y en los coches aparcados. Problemas similares se registraron en San Sebastián debido a las olas de 11 metros de altura. Frente al cabo Vilán una boya registro olas de

22m de altura, estas olas extremas no se había registrado nunca.

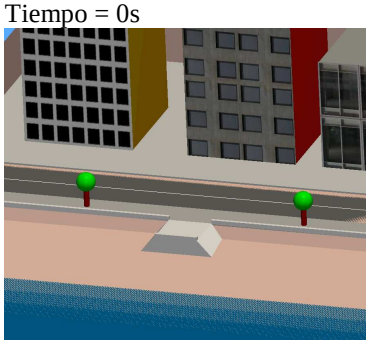
FIG. 8. Efecto de tormentas en la costa de A Coruña, 2008 (fuente: “La Voz de Galicia”).

DualSPHysics puede ser utilizado para hacer frente a problemas de ingeniería de la vida real, siendo el campo de protección de la costa un candidato adecuado para ser estudiado. Ya que, se consideran grandes dominios y el tiempo de ejecución de cómputo puede tardar días en una sola CPU.

La Figura 9 muestra una simulación de una situación realista, el malecón, el pavimento, la calle, árboles, edificios y la playa fueron implementados para interactuar con una gran ola. Un sistema de 150m x 100m x 20m fue simulado con una distancia entre las partículas de 0.3m. Esto implica una simulación de un millón de partículas para el contorno y de 5.183.904 partículas para el fluido. Donde 20 segundos de tiempo real se ejecuto en una Tesla C1060, con un rendimiento de 2,52 pasos de cómputo por segundo. Por otra parte, el espacio de memoria máximo presente en estas GPUs hace posible una simulación con 20 millones de partículas.

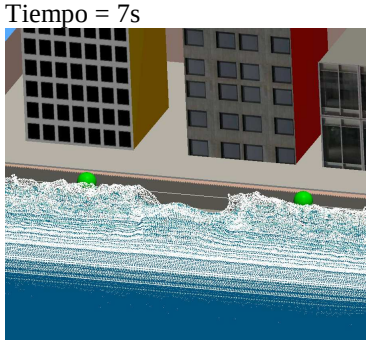
Diferentes instantes de la simulación se muestran en la siguiente figura. El modelo se ocupa de la interacción ola-estructura, proporcionando información valiosa acerca de cuáles son las zonas inundadas y como. De esta forma, con el uso de modelos SPH, vamos a poder elaborar mapas de riesgo en las zonas costeras.

Tiempo = 0s

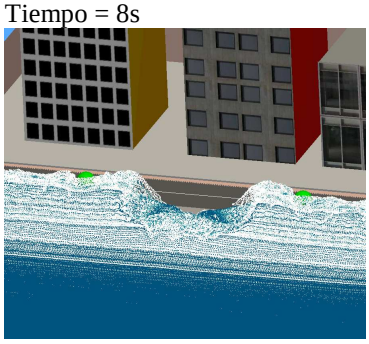


Tiempo = 5s

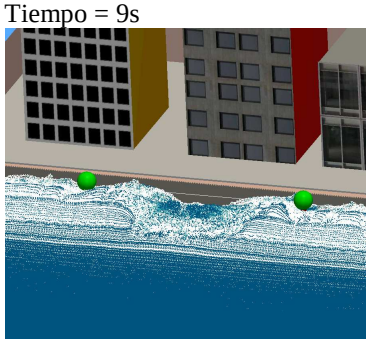
Tiempo = 7s



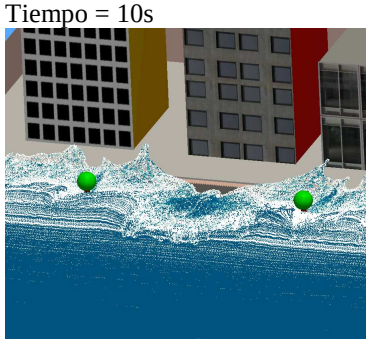
Tiempo = 8s



Tiempo = 9s



Tiempo = 10s



Tiempo = 11s

Tiempo = 16s

Tiempo = 18s

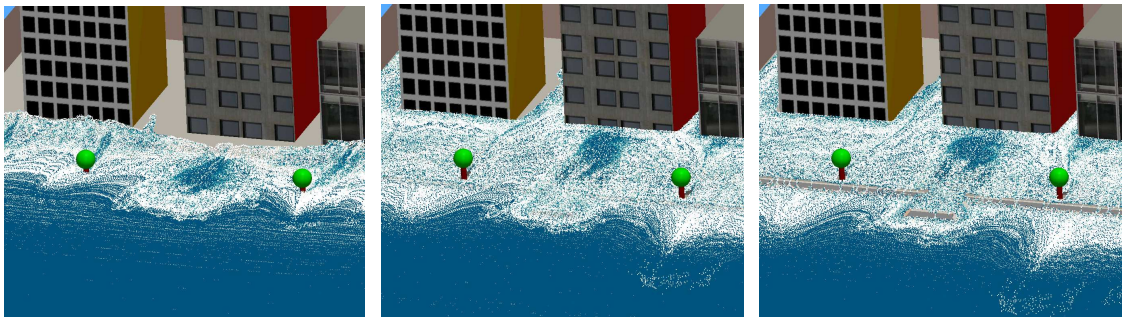


FIG. 9. Diferentes instantes de la simulación de grandes olas con el paseo marítimo. En el siguiente enlace http://ephyslab.uvigo.es/index.php/eng/dual_sphysics/ puede accederse a la animación completa.

6 Conclusiones

La técnica SPH descrita anteriormente es el fundamento del código SPHysics (www.sphysics.org) desarrollado en estos últimos años (Gómez-Gesteira et al. 2009). Este modelo se ha utilizado para reproducir diferentes experimentos y así conseguir resultados que se ajusten a los casos reales.

Se ha probado que el método SPHysics puede reproducir tanto cualitativa como cuantitativamente fenómenos de interacción entre estructuras y olas. Así, se han estudiado fenómenos de rebase de olas sobre una plataforma (Gómez-Gesteira et al. 2005), interacción entre fluidos y propagación de oleaje (Crespo et al. 2008a; Crespo et al. 2008b y Narayanaswamy et al. 2009), interacción 3D entre una ola y una estructura costera (Gómez-Gesteira y Dalrymple 2004). Del mismo modo se ha aplicado al estudio de diferentes escenarios de protección de estructuras costeras frente a olas muy grandes (Crespo et al. 2007a).

Mediante este estudio queda de manifiesto la idoneidad del modelo para reproducir la interacción entre el fluido y las estructuras fijas y la capacidad de tratar fluidos no lineales. Es decir, en este caso, el flujo de agua se rompe rodeando los huecos entre los edificios para más tarde volverse a reunir, ésta no conectividad del fluido tridimensional es reproducida perfectamente por este método lagrangiano (tal y como se demostró en Crespo et al. 2007 y Crespo et al. 2008b).

Muchos problemas de ingeniería e incluso la simulación de escenarios reales necesitan una computación masiva, donde sólo la simulación de millones de partículas nos permite analizar más en detalle los fenómenos físicos que se desarrollan en el sistema. La versión actual de SPHysics es capaz de aprovechar el alto rendimiento que nos proporcionan las GPU.

Esta tecnología nos permite simular experimentos numéricos con un gran número de partículas en lugar de reproducir experiencias reales que son caras y en algunas ocasiones imposibles de llevarse a cabo con modelos a escala y, por otro lado, proporciona información adicional que no se puede obtener de la observación experimental y así entender los procesos físicos envueltos en el fenómeno.

Referencias:

Crespo, A. J. C., Gómez-Gesteira, M. y Dalrymple, R. A. 2007: 3D SPH simulation of large waves mitigation with a dike. *Journal of Hydraulic Research*, 45, 5, 631- 642.

Crespo, A. J. C., Gómez-Gesteira, M. y Dalrymple, R. A. 2007: Boundary Conditions Generated by Dynamic Particles in SPH Methods. *CMC:*

Computers, Materials, & Continua, 5, 3, 173-184.

Crespo, A. J. C., Gómez-Gesteira, M. y Dalrymple, R. A. 2008: Modeling Dam Break Behavior over a Wet Bed by a SPH Technique. *Journal of Waterway, Port, Coastal, and Ocean Engineering*, 134, 6, 313-320.

Crespo, A. J. C., Gómez-Gesteira, M., Carracedo, P. y Dalrymple, R. A. 2008: Hybridation of generation propagation models and SPH model to study severe sea states in Galician Coast. *Journal of Marine Systems.* 72, 135-144.

Crespo, A. J. C., Marongiu, J. C., Parkinson, E., Gómez-Gesteira, M., Dominguez, J. M. 2009: High Performance of SPH Codes: Best approaches for efficient parallelization on GPU computing. Proc. IVth Int. SPHERIC Workshop (Nantes), pp. 69-76.

Dalrymple, R. A. and Rogers, B. D. 2006: Numerical Modeling of Water Waves with the SPH Method. *Coastal Engineering*, 53, 2, 141-147.

Dominguez, J. M., Crespo, A. J. C., Gómez-Gesteira, M. and Marongiu J.C., Neighbour lists in Smoothed Particle Hydrodynamics. *International Journal for Numerical Methods in Fluids*, in press.

Gómez-Gesteira, M. and Dalrymple, R. A. 2004: Using a 3D SPH Method for Wave Impact on a Tall Structure. *Journal of Waterway, Port, Coastal, and Ocean Engineering*. 130, 2, 63-69.

Gómez-Gesteira, M., D. Cerqueiro, A. J. C. Crespo y R. A. Dalrymple. 2005: Green water overtopping analyzed with a SPH model. *Ocean Engineering*. 32, 223-238.

Gómez-Gesteira, M., Rogers, B. D., Violeau, D., Grassa, J. M. y Crespo, A.J.C. 2010: SPH for free-surface flows. *Journal of Hydraulic Research*. 48 (Extra issue), 3-5.

Gómez-Gesteira, M., Rogers, B. D., Dalrymple, R. A., Crespo, A. J. C. and Narayanaswamy, M. 2010: User Guide for the SPHysics Code v2.0.

Gingold, R. A. and Monaghan, J.J. 1977: Smoothed particle hydrodynamics: theory and application to non- spherical stars. *Mon. Not. R. Astr. Soc.*, 181, 375-389.

Lucy, L. 1977: A numerical approach to the testing of fusion process. *Journal Astronomical.* 82, 1013-1024.

Narayanaswamy, M. S., Crespo, A. J. C., Gómez-Gesteira, M. y Dalrymple, R. A. 2010: SPHysics-Funwave hybrid model for coastal wave propagation. *Journal of Hydraulic Research.* 48 (Extra issue), 63-71