Asociación Argentina
de Mecánica Computacional

Mecánica Computacional Vol XXVI, pp.992-999
Sergio A. Elaskar, Elvio A. Pilotta, Germán A. Torres (Eds.)
Córdoba, Argentina, Octubre 2007

TÉCNICAS ÁRBOL EN INTERACCIONES DE N PARTÍCULAS ORIENTADAS A ELEMENTOS DE BORDE

Jorge D'Elía, Laura Battaglia y Mario A. Storti

Centro Internacional de Métodos Computacionales en Ingeniería (CIMEC), Instituto de Desarrollo Tecnológico para la Industria Química (INTEC), Universidad Nacional del Litoral - CONICET , Güemes 3450, 3000-Santa Fe, Argentina, e-mail: lbattaglia@ceride.gov.ar (mstorti,jdelia)@intec.unl.edu.ar , web page: http://www.cimec.org.ar

Palabras clave: árbol descendente de Barnes-Hut, técnicas rápidas para n partículas, curva de llenado del espacio, curva de Peano-Hilbert, algoritmo de Butz, método de elementos de borde.

Resumen.

La colocación por puntos es una técnica estándar en el método de los elementos de borde (*Boundary Element Method*, BEM), con la cual se obtiene una solución numérica aproximada para diversos tipos de ecuaciones integrales. En particular estamos interesados en el caso de un núcleo (o función de Green) débilmente singular, donde el dominio está inmerso en el espacio euclídeo tridimensional, y cuyo borde es aproximado por una superficie poliédrica formada por triángulos planos. Cuando el número de elementos crece el costo computacional se encarece relativamente rápido debido a la naturaleza no-local de la función de Green, la cual conduce, a nivel discreto, a una matriz del sistema densa. Para mitigar el costo computacional se han propuesto en la literatura diversas variantes. Entre otras, están las de combinar BEM con las técnicas rápidas de n partículas. Dos variantes populares comprenden las de tipo árbol (quadtree/octtree) y las de multipolos rápidos. Ambas familias se basan en una subdivisión recursiva del espacio en hipercubos combinada con una estructura de datos de tipo árbol. En este trabajo se describen las primeras experiencias en el desarrollo de algoritmos rápidos para simulaciones de n partículas, todas contra todas, en una variante jerárquica (o en árbol) orientados a su adaptación en elementos de borde.

1. INTRODUCCIÓN

En los métodos numéricos que involucran interacciones entre n partículas (Marzouk and Ghoniem, 2005; Bowers et al., 2007), todas contra todas, la interacción involucra típicamente alguna función de Green con ley inversa de la distancia, en donde además cada partícula tiene asociado un número constante de atributos geométricos (e.g. posición y área) y físicos (e.g. masa, potenciales dipolar y monopolar). Una taxonomía posible según la forma en que se hace el cálculo da lugar a las interacciones Amor (1997): Partícula-Partícula (PP), Partícula-Multicelda (PM), Partícula-Partícula-Partícula-Multicelda (P3M) y Jerárquicas, o en árbol. Los algoritmos PP son los más clásicos dados básicamente por un par de lazos anidados de 1 a n, con el consecuente costo computacional $O(n^2)$, dando lugar, en principio, a matrices de influencia densas o llenas, que es el caso básico en los métodos de elementos de borde (BEM). Los algoritmos PM y P3M son mejoras intermedias algo más usadas en física de plasmas. En cambio, los algoritmos jerárquicos, o en árbol, fueron diseñados para reducir el costo computacional tradicional $O(n^2)$ a $O(n \log n)$ usando diversas estratagemas, e incluyen los de Appel (1985), Barnes and Hut (1986), Xue (1998), y los multipolos rápidos (Fast Multipole Methods, FMM) de Greengard (1988). Notar que originalmente el FMM fue erróneamente catalogado como $O(n)$, e.g. ver los análisis de Aluru (1996), Darve (2001), error que llamativamente persiste en parte de la bibliografía.

La utilidad de los métodos jerárquicos se evidencia cuando se los entrelaza con los métodos iterativos usados para resolver aproximadamente el Sistema Discreto de Ecuaciones (SDE) asociado a balances en las interacciones entre las n partículas. En los algoritmos PP se requiere almacenar toda matriz del sistema en la memoria RAM, o bien recalcularla cada vez que se la necesite. Ambas alternativas son muy restrictivas pues, en la primera el costo de memoria RAM también es $O(n^2)$, y en la segunda lo es por recalcular una matriz que es densa. Esta problemática nunca se presenta cuando se usan las matrices banda obtenidas, por ejemplo, por elementos finitos (FEM) en forma estándar. Otra consecuencia es que el número de grados de libertad usados en BEM tiende a ser mucho menor que en FEM, lo cual dificulta un uso combinado de ambos métodos.

En este trabajo resumimos las primeras etapas en una primera implementación de los algoritmos rápidos para BEM, basada en el árbol recursivo descendente auto-adaptativo de Barnes-Hut. La orientación final es el BEM de bajo orden en la formulación de Morino para flujos tridimensionales. Las partículas estarán ubicadas en los centroides de los paneles de la malla BEM y tendrán asociadas el área y los potenciales dipolar y monopolar de cada panel. La implementación prevé además el ordenamiento de Peano-Hilbert, de utilidad en la ulterior versión paralela del algoritmo, ver Amor et al. (2001a,b). El ordenamiento de Peano-Hilbert se hace mediante dos alternativas, la primera es la tabular dada por la taxonomía de Campbell et al. (2007), mientras que la segunda se basa en el algoritmo de Butz (1971), retomado por Lawder (2000).

2. ARBOL DESCENDENTE DE BARNES-HUT AUTO-ADAPTATIVO

Los algoritmos jerárquicos construyen primero una estructura de datos de tipo árbol (Storti et al., 2007) para la representación jerárquica del espacio $\mathbb{R}^d$ en hipercubos y luego se calculan las interacciones recorriendo el árbol, posiblemente en forma parcial. Cada partícula sólo interactúa con una parte del árbol, pues las secciones de la nube de puntos que están más alejadas de la partícula bajo observación interactúan en los niveles más bajos del árbol, esto es, más cerca de la raíz, mientras que las secciones más cercanas interactúan en los niveles más altos

del árbol, es decir más cerca de las hojas. La construcción del árbol, naturalmente recursiva, se puede hacer en forma descendente, desde la raíz hacia las hojas, como fue propuesto en la versión original de Barnes and Hut (1986) (BH), a su vez una mejora al de Appel (1985), o en forma ascendente, desde las hojas hasta la raíz, en la versión de Xue (1998), cada una con sus ventajas y desventajas. El algoritmo recursivo de BH es inherentemente adaptativo pues no hace ninguna suposición sobre la distribución espacial de las partículas en el hipercubo raíz. Aquí seguiremos el de BH mejorado por Amor (1997), en donde las partículas se van ingresando de a una. La construcción del árbol T puede resumirse con el siguiente pseudo código, que es una adaptación del dado por Demmel (2007),

```
procedure hacer-arbol (X,T) {
  n = size (X,dim=1)  !numero de particulas
  d = size (X,dim=2)  !dimension del espacio
  inicia-arbol (X,T,raiz)
  for i = 1 to n do
    ingresa-particula (X,T,raiz,i)
}
```

donde $\mathbf{X} \in \mathbb{R}^{n \times d}$ es la matriz de coordenadas de los puntos en $\mathbb{R}^d$, mientras que n y d son el número de partículas y de dimensiones del espacio, respectivamente. Cada nueva partícula añadida va recorriendo el árbol actual desde la raíz hasta el vértice del nivel más bajo que la puede contener, modificando el baricentro y la masa de los vértices por los cuales va pasando. Esta mejora de ir actualizando al pasar por los vértices tiene la ventaja parcial de evitar una etapa adicional para calcularlos. Si (i) el vértice destino está vacío entonces la partícula se añade directamente; (ii) si ya estaba ocupado por otra, entonces se agregan r vértices hijos (con $r = 8$ para *octrees* en $\mathbb{R}^3$ y $r = 4$ para *quadtrees* en $\mathbb{R}^2$), se divide su celda espacial (un hipercubo de un cierto tamaño) en r hipercubos iguales, que son numerados en forma conveniente, y se reubican las dos partículas (la pre-existente y la recién llegada) en los hijos que les correspondan a cada una (incluso podrían coincidir en un mismo hijo); (iii) finalmente, si hay más de dos hijos, entonces se identifica la celda donde hay que colocarla. A la construcción recursiva descendente la resumimos con el siguiente pseudo código adaptado de Demmel (2007),

```
procedure ingresa-particula (X,T,m,i) {
  !intenta agregar la particula i en el vertice m
  !por construccion cada hoja tendra 1 o 0 particula
  select case (T (m).nroparticulas)
  case (0)  !es una hoja sin ocupar
    almacenar la particula i en el vertice m
  case (1)  !es una hoja ya ocupada
    agregar r-hijos dividiendo el hipercubo del vertice m
    mover la particula preexistente al hijo correspondiente
    actualizar baricentros, masa y potencial
    identificar el hijo c donde hay que colocar la particula i
    ingresa-particula (X,T,c,i)
  case (2:) !tiene 2 o mas particulas
    identificar el hijo c donde hay que colocar la particula i
    ingresa-particula (X,T,c,i)
  end select
}
```

Con respecto a cómo se hace la actualización del baricentro y del potencial de los vértices por donde va pasando la nueva partícula ingresada en el árbol, supongamos que en una dada

etapa tanto el potencial W_u (dipolar o monopolar) del vértice u como su baricentro $\mathbf{x}_u$ están dados por

$$\begin{aligned} W_u &= \sum_{i=1}^{U} W_i \ ; \\ \mathbf{x}_u &= \frac{1}{W_u} \sum_{i=1}^{U} W_i \mathbf{x}_i \ ; \end{aligned} \tag{1}$$

donde U es el número total de partículas contenidas en el subárbol que empieza en el vértice u. Al añadir una nueva partícula i a la celda, la nueva posición del baricentro y su potencial se actualizan con

$$\begin{aligned} W_{u+1} &= W_u + W_i \ ; \\ \mathbf{x}_{u+1} &= \frac{1}{W_{u+1}} (\mathbf{x}_u W_u + \mathbf{x}_i W_i) \ ; \end{aligned} \tag{2}$$

donde W_i y $\mathbf{x}_i$ es el potencial y la posición, respectivamente, de la partícula añadida al vértice. La construcción del árbol finaliza cuando se ingresaron las n partículas, donde cada hoja tiene 0 o 1 partícula.

Una vez construido el árbol interesa calcular el producto matriz vector $\mathbf{q} = \mathbf{A}\mathbf{p}$, donde $\mathbf{A}$ es una matriz (densa) de influencia entre las n partículas. Este producto es típico de los métodos iterativos tipo gradientes conjugados (Ashby et al., 1990) o los de tipo GMRes (Saad and Schultz, 1986), donde cada entrada

$$q_i = \sum_{j=1}^{n} A_{ij} p_j \ ; \tag{3}$$

representa, en BEM, la influencia total de las n partículas sobre la partícula genérica i bajo observación, incluyendo su propia autoinfluencia. En lugar de calcular el vector $\mathbf{q}$ en la forma tradicional con un doble lazo de 1 a n lo haremos en forma recursiva. Empezando desde la raíz del árbol, vamos acumulando la influencia que ejercen los vértices del árbol sobre la partícula i en observación y si, en cada contribución, el baricentro del vértice genérico r estuviera lo suficientemente lejos, entonces el subárbol entero con raíz en ese vértice es aproximado por alguna expansión multipolar conveniente y no se sigue bajando en el subárbol, en caso contrario se repite el proceso con cada uno de los hijos del vértice r llegando inclusive a las hojas si fuera necesario.

Se desprende la necesidad de un test para decidir cuándo una celda está lo suficientemente lejos, test que se denomina como Criterio de Aceptación Multipolar (CAM). Hay varias propuestas, todas basadas en consideraciones geométricas vinculadas con el diámetro de la celda y las distancias relativas. La de BH considera que una celda está bien separada si $D/L < \theta$, donde D es algún diámetro de la celda, $L = ||\mathbf{x} - \mathbf{y}||_2$ es la distancia euclídea entre el punto de observación $\mathbf{x}$ y el baricentro de la celda $\mathbf{y}$, y θ es un cierto valor umbral prefijado. Un umbral bajo sugiere que una partícula tiene que estar muy distante de la celda fuente antes de aceptar la expansión multipolar, con lo cual el costo computacional se encarece pues el árbol tiene que ser atravesado hasta sus niveles más profundos, haciendo que se calculen un mayor número de interacciones directas entre partículas, y en el límite $\theta \to 0$ se recupera el caso todas contra todas, con su tradicional costo $O(n^2)$. En cambio cuando $0 \ll \theta \leq 1$ hay menos interacciones

directas. Por experimentos numéricos se ha encontrado que θ escala como $O(n^{-3})$ en $\mathbb{R}^3$. En astrofísica se suele usar $0.7 \leq \theta \leq 1$, con aproximación multipolar hasta el cuadripolo. En definitiva, al producto matriz vector $\mathbf{q} = \mathbf{Ap}$ lo resumimos con el siguiente pseudo código, también adaptado de Demmel (2007),

```
function q = daxpy (X, T, p) {
  n = length (p) ; raiz = root-arbol (T)
  q = 0
  for i = 1 to n do
    q (i) = influencia (X,T,raiz,i)
}
```

```
function f = influencia (X,T,r,i) {
  !influencia f sobre la particula i causada por
  !subarbol de raiz r
  f = 0
  x = X (i,:)          ;  y = T (r).baricentro
  D = T (r).diametro   ;  L = ||x - y||_2
  select case (T (r).nroparticulas)
  case (1)
    f = aproximacion-multipolar (x,y)
  case (2:)
    if (D/L < theta) then
      f = aproximacion-multipolar (x,y)
    else
      forall (hijos c de r) do f = f + infuencia (X,T,c,i)
    end if
  end select
}
```

Notar que también es un algoritmo recursivo que va recorriendo el árbol de datos posiblemnete en forma parcial pues no necesariamente debe llegar hasta todas sus hojas.

3. CURVAS DE LLENADO DEL ESPACIO

La descomposición en árbol r-ario de las partículas involucra un cierto ordenamiento unidimensional de los hipercubos hijos. Por otra parte, para reducir las comunicaciones en un algoritmo paralelo basado en el paradigma de paralelismo de datos, es conveniente preservar lo mejor posible la proximidad física de las partículas (Jin and Crummey, 2005; Hamilton and Chaplin, 2007; Anderson, 1996; Dennis, 2003; Waltz et al., 2002). Para tal fin, se opta por numerarlas usando una curva de llenado del espacio $\mathbb{R}^d$. Estas curvas tienen al menos dos propiedades interesantes: (i) dos puntos adyacentes en la curva son también adyacentes en $\mathbb{R}^1$, y (ii) las subregiones del espacio $\mathbb{R}^d$ definidas por segmentos continuos de la curva son conexas.

Las curvas de llenado del espacio por construcción preservan en $\mathbb{R}^1$ la proximidad espacial de los puntos en $\mathbb{R}^d$, pero el grado con que lo hacen depende de la variante empleada. Entre otras, son conocidas la de Morton, la de Gray o la de Peano-Hilbert (PH), e.g. ver Campbell et al. (2007), Lawder (2000). En lo que sigue sólo nos concentraremos en la de PH. Esta curva es el límite de una poligonal que recorre el hiper-cubo unitario de tal modo de asegurar un mapeo continuo entre $\mathbb{R}^1$ y $\mathbb{R}^d$. El uso de esta poligonal facilita una partición de datos que debe hacerse en los algoritmos basados en el paradigma de paralelismo en los datos.

Por ejemplo, las dos primeras etapas de un proceso infinito cuando $d = 2$ se pueden resumir como sigue, ver Fig. 1. En el primer paso, el cuadrado unitario se divide en 4 sub-cuadrados,

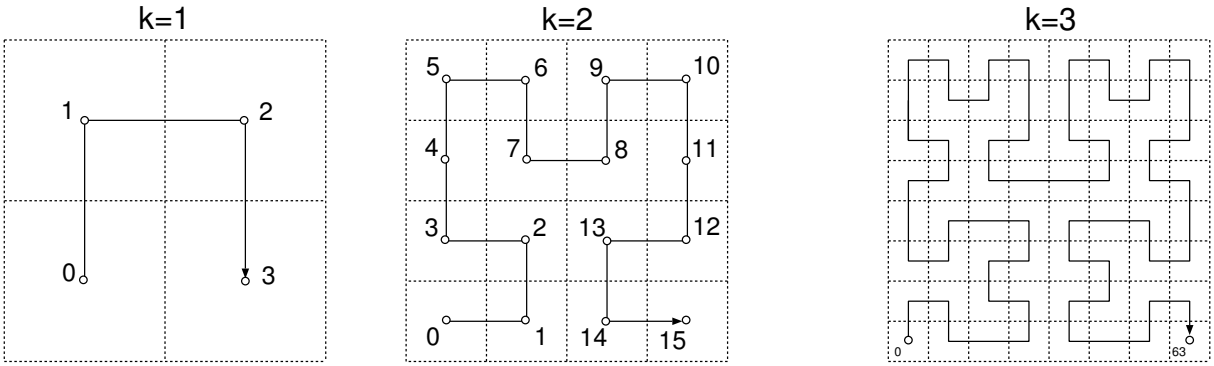

Figura 1: Poligonales de Peano-Hilbert de orden $k = 1, 2, 3$ en 2 dimensiones ($d = 2$).

los cuales son numerados de modo tal que cualquier par de cuadrados consecutivos comparten una arista. Este ordenamiento se representa mediante la poligonal de primer orden ($k = 1$ en la figura) que pasa por los centroides de los sub-cuadrados. En el segundo paso, cada subcuadrado es a su vez subdividido en otros 4 subcuadrados, en donde los sub-cuadrados dentro del primer y último cuadrado de la primera etapa son reordenados de modo tal de asegurar la propiedad de adyacencia, dando lugar a la poligonal de segundo orden ($k = 2$ en la figura). En principio, el proceso puede continuar indefinidamente, pero en las aplicaciones prácticas es finalizado después de k-etapas y se tiene la poligonal de llenado del espacio de Peano-Hilbert de orden k, la cual pasa a través de 2^{kd} subcuadrados cuyos centroides pueden ser asimilados a puntos en un espacio de granularidad finita.

Buena parte de los algoritmos involucrados para tal fin se basan en uso de reglas de construcción de diagramas de estado auxiliares para el cálculo de los mapeos, o bien de tablas con doble entrada, e.g. las de Campbell et al. (2007). Empero, como observa Lawder (2000), las reglas suelen ser incompletas, requiriendo un proceso de prueba-error para completarlas y/o depurarlas, mientras que el requerimiento de memoria para los diagramas de estado crece exponencialmente con la dimensión d del espacio considerado, y han mostrado ser manejables hasta ocho dimensiones.

Como alternativa, el algoritmo de Butz (1971) prescinde del uso tanto de tablas como de los digrafos de estado finito, y permite construir la poligonal de Peano-Hilbert en el hipercubo unitario en $\mathbb{R}^d$ para cualquier valor de k. Las presentaciones de Butz, a través de sus tablas I y II, y la de Lawder (2000) son perfectamente suficientes para su codificación.

4. CONCLUSIONES

Los algoritmos jerárquicos tanto tipo Barnes and Hut (1986) como también los multipolos rápidos FMM, son muy conocidos en astrofísica desde hace tiempo, pero sólo recientemente están empezando a ser empleados en técnicas por elementos de borde. Por otra parte, para reducir las comunicaciones en un procesamiento en paralelo, interesa un buen ordenamiento de los hipercubos asociados a cada uno de los vértices. Una instancia para tal fin, orientada al paradigma del paralelismo de datos, es numerarlos usando alguna de las poligonales de llenado del espacio. La poligonal de llenado del espacio de Peano-Hilbert obtenida por la taxonomía de Campbell et al. (2007) involucra el uso de dos tablas de cuatro entradas cada una en $\mathbb{R}^2$ y de veinticuatro entradas cada una en $\mathbb{R}^3$, con la ventaja de una implementación más sencilla, pero con el inconveniente de estar restringido a esas dimensiones. En cambio, el algoritmo de Butz (1971) es quizás algo más complicado de programar pero es sistemático, libre de casos

especiales y es más general pues fue pensado para el espacio euclídeo $\mathbb{R}^d$. En el código en construcción se ha implementado hasta ahora la fase de construcción del árbol, su posterior recorrido para obtener el producto matriz-vector y la generación de la poligonal de llenado del espacio en dos variantes, tanto mediante las tablas con doble entrada de Campbell et al. como con el algoritmo sistemático de Butz. Como trabajo futuro se prevé incorporar un solver iterativo en un problema resuelto por elementos de borde.

Agradecimientos

Este trabajo ha sido financiado por el Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET, Argentina, proyectos PIP 02552/00, PIP 5271/05), Universidad Nacional del Litoral (UNL, Argentina, proyecto CAI+D 2005-10-64) y la Agencia Nacional de Promoción Científica y Tecnológica (ANCyT, Argentina, proyectos PICT 12-14573/2003, PME 209/2003), y ha sido parcialmente realizado con los recursos del Free Software Foundation/GNU-Project, tales como GNU/Linux OS, compilador GNU/GFortran, GNU /Octave, así como otros recursos Open Source, tales como MPICH, ADAPTOR y Tgif.

REFERENCIAS

Aluru S. Greengard's n-body algorithm is not O (n). *SIAM Journal Scientific Computing*, 17:773–776, 1996.

Amor M. *Divide and Conquer Algorithms on Processor Meshes*. Ph.D. thesis, Dpto de Electrónica y Sistemas, Universidad de Santiago de Compostela, 1997.

Amor M., Argüello F., López J., Plata O., and Zapata E.L. A data parallel formulation of the Barnes-Hut method for n-body simulations. In S.T. et al., editor, *PARA 2000, LNCS 1947*, pages 342–349. Springer-Verlag, Berlin, 2001a.

Amor M., Argüello F., López J., Plata O., and Zapata E.L. A data parallel formulation of the divide and conquer. *The Computer Journal*, 44:303–320, 2001b.

Anderson R.J. Tree data structures for n-body simulation. In *37th Annual Symposium on Foundations of Computer Science*, pages 224–233. IEEE Computer Society, 1996.

Appel A.W. An efficient program for many-body simulations. *SIAM J. Sci. Stat. Comput.*, 6(1):85–103, 1985.

Ashby S.F., Manteuffel T.A., and Saylor P.E. A taxonomy for conjugate gradient methods. *SIAM Journal on Numerical Analysis*, 27(6):1542–1568, 1990.

Barnes J. and Hut P. A hierarchical O (n log n) force-calculation algorithm. *Nature*, 324:446–449, 1986.

Bowers K.J., Dror R.O., and David E. Shaw D.E. Zonal methods for the parallel execution of range-limited n-body simulations. *Journal of Computational Physics*, 221:303–329, 2007.

Butz A. Alternative algorithm of Hilbert space-filling curve. *IEEE Trans. On Computers*, 20:424–426, 1971.

Campbell P.M., Devine K.D., Flaherty J.E., Gervasio L.G., and Teresco J.D. Dynamic octree load balancing using space-filling curves. Technical report, Los Alamos, 2007.

Darve E. The Fast Multipole Method I: Error analysis and asymptotic complexity. *SIAM Journal on Numerical Analysis*, 38(1):98–128, 2001.

Demmel D. Fast hierarchical methods for the n-body problem. parts 1 and 2. 2007. http://www.cs.berkeley.edu/ demmel/ cs267/lecture26/ lecture26.html/.

Dennis J.M. Partitioning with space-filling curves on the cubed-sphere. In *17th International Symposium on Parallel and Distributed Processing IPDPS '03*, pages 1–6. IEEE Computer Society, 2003.

Greengard L. *The rapid evaluation of potential fields in particle systems*. MIT Press, 1988.

Hamilton C.H. and Chaplin A.R. Compact Hilbert indices for multi-dimensional data. In *First International Conference on Complex, Intelligent and Software Intensive Systems, CISIS '07*, pages 1–8. IEEE Computer Society, 2007.

Jin G. and Crummey J.M. Using space-filling curves for computation reordering. In *6th Annual Symposium*, pages 1–9. Los Alamos Computer Science Institute, published on CD, 2005.

Lawder J.K. Calculation of mapping between one and n-dimensional values using the Hilbert space-filling curve. Technical Report JL1/00:1-13, UK, London, 2000.

Marzouk Y.M. and Ghoniem A.F. K-means clustering for optimal partitioning and dynamic load balancing of parallel hierarchical n-body simulations. *Journal of Computational Physics*, 207:493–528, 2005.

Saad Y. and Schultz M. Generalized minimum residual method (GMRES). *SIAM J. Sci. Statist. Comput.*, 7:856–869, 1986.

Storti M., D'Elía J., Paz R., and Dalcín L. Algoritmos y Estructuras de Datos (notas y transparencias en formato pdf). Cátedra AED, Dpto de Informática, Facultad de Ingeniería y Ciencias Hídricas (FICH), Universidad Nacional del Litoral, 2007.

Waltz J., Page G.L., Milder S.D., Wallin J., and Antunes A. A performance comparison of tree data structures for n-body simulation. *Journal of Computational Physics*, 178:1–14, 2002.

Xue G. A cost optimal parallel algorithm for computing force field in n-body simulations. In Hsu and Kao, editors, *COCOON 98, LNCS 1949*, pages 95–104. Springer Verlag, Berlin, 1998.