Unidad 2
Implementación didáctica del algoritmo de árbol jerárquico octal

La evaluación, a un tiempo dado, de las interacciones del problema de $N-$cuerpos vía el algoritmo de árbol jerárquico octal involucra tres pasos: (i) construcción del árbol, (ii) cómputo del desarrollo multipolar del sistema ("adornado del árbol") y, finalmente, (iii) evaluación de las fuerzas. Nuestra implementación, en FORTRAN 77, del algoritmo desarrolla estos pasos a través de la interconección de un conjunto de subrutinas y funciones como se ilustra en la figura 1. Las secciones discuten, brevemente, tal implementación.

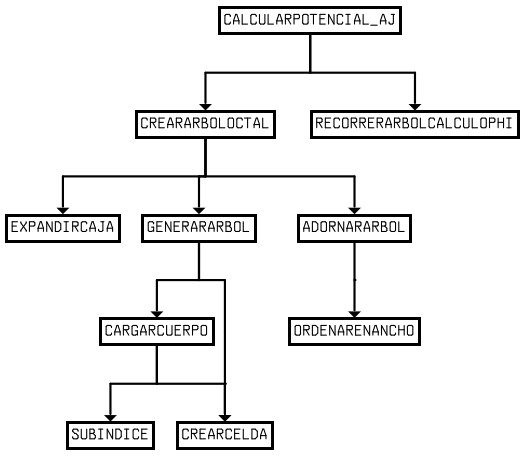

Figura 1. Esquema de las llamadas de las subrutinas y funciones para el cálculo de las fuerzas a un tiempo dado por el método del árbol jerárquico.

Construcción del árbol. Como primer paso de la construcción del árbol, se necesita una división del espacio que permita posteriormente discernir las partículas próximas de las lejanas a una dada partícula. Para ello comenzamos con una *celda* cúbica suficientemente grande para contener todo el sistema de partículas. Las partículas son entonces colocadas (*cargadas*) en esta celda *raíz* (o *root*). Si dos partículas caen en la misma celda (lo que sucederá tan pronto como la segunda partícula se cargada en la celda raíz), la celda se divide en subceldas teniendo éstas exactamente la mitad de largo, ancho y alto que la celda que las genera. Esto significa que, en tres dimensiones, la celda original es dividida en ocho subceldas (de aquí el calificativo de *octal* del árbol)[1]. Si dos partículas caen aún en una misma subcelda, ésta es dividida recursivamente hasta que las partículas estén ubicadas en diferentes compartimientos. Entonces, la segunda partícula es cargada y el mismo procedimiento comienza nuevamente desde la celda raíz. Cuando todas las partículas N hayan sido cargadas, el espacio que ocupa el sistema habrá sido particionado en un determinado número de celdas cúbicas de diferentes tamaños, con al menos una partícula por celda (puede estar vacía). La estructura de árbol proviene de asignar, en un dado nivel del proceso de división de la celda original, "hojas" y "ramas": una hoja apunta a una partícula y una rama a una celda que ha sido subdividida (celdas vacías son ignoradas).

En la implementación del algoritmo es la subrutina `creararboloctal` quien se encarga de generar el volumen que contiene a todo el sistema, a través de la subrutina `expandircaja` y de generar la estructura del árbol y cargar los cuerpos, a través de la subrutina `generararbol`. La subrutina `expandircaja` calcula de forma simple la dimensión de la celda más grande como potencia de 2 que contiene a todo el sistema. La subrutina `generararbol` origina la estructura del árbol al ir cargando uno a uno los cuerpos. Para cada cuerpo, que será una hoja en el árbol, un índice es utilizado a modo de etiqueta para identificarlo, si `ncuerpos` es el número de partículas de la simulación tal índice sera un número entero positivo $i = 1, 2, \ldots, ncuerpos$ (En la implementación un número máximo de partículas que puede manejar el programa está definido por el parámetro `MAXCUERPOS`). Este índice provee la conexión con las cantidades físicas asociadas a las partículas, tales como la posición y la masa. Asimismo cada celda debe ser asociada a un índice que la identifique como celda, tal índice en nuestra implementación comienza a contarse a partir del número entero positivo `INICELDAS=MAXCUERPOS+1`. La función `crearcelda` inicializa apropiadamente este índice y genera un arreglo `subceldas(i,k)` que indica el estado de cada una de las $k = 1, \ldots, 8$ subceldas descendientes de la celda $i-$ésima, esto es, si esta vacía o ocupada, y si está ocupada quien la ocupa: otra celda, o un cuerpo. Usando

[1] En dos dimensiones, una celda generará cuatro subceldas.

esta información junto con el resultado de una función `subindice` que indica la subcelda que ocupa el cuerpo respecto de una celda en un dado nivel de construcción, cada cuerpo es cargado adecuadamente en una hoja del árbol a través de la subrutina `cargarcuerpo`.

Todas las subrutinas incluyen un archivo, `ncuerposvars.inc` que contiene la definición de los parámetros necesarios, así como la declaración de las variables utilizadas, y otros valores necesarios para el programa. En particular, la variable lógica `VERRECORRIDO` definida como `.true.` nos permitirá ver la construcción del árbol.

Adornando del árbol. Una vez que la estructura de árbol está construida, además de las cantidades físicas de asociadas a las partículas, es necesario que cantidades semejantes sean calculadas y guardadas para las celdas (ramas), debido a que esta información será necesaria para el cálculo de la fuerza. Este "proceso de adornar el árbol" puede hacerse eficientemente propagando la información hacia abajo del árbol desde las hojas (partículas) hacia la raíz. Así, la masa, posición del centro de masa y momentos cuadrupolares [2] de una celda son obtenidos a partir del conocimiento de estas cantidades en sus celdas descendientes. Este procedimiento mantiene el esfuerzo de cómputo al mínimo.

En nuestra implementación, cuando la estructura de árbol está construida, la subrutina `creararboloctal` llama a la subrutina `adornararbol` para efectuar las cálculos descritos. Dado que es necesario ordenar las celdas de las más pequeñas a las más grandes para proceder con el cálculo, es invocada en primer lugar una subrutina `ordenarenarenancho` para hacer ésto (En realidad, esta subrutina genera una orden en tamaño decreciente, así que el ordenamiento contenido en el vector `indceldas` generado por la subrutina debe utilizarse inversamente).

Cálculo de la fuerza. Una vez que el árbol ha sido construido (y "adornado"), esta estructura es ahora usada pora realizar el cálculo de la fuerza (en rigor, la fuerza por unidad de masa, esto es, la aceleración) sobre cada partícula del sistema. La idea básica es incluir las contribuciones de las partículas cercanas como una interacción directa partícula–partícula, mientras que las contribuciones de partículas lejanas es tomada en cuenta a través de las celdas que las incluyen, esto es, como un grupo de partículas. Este esquema tiene la ventaja de retener el concepto de interacción partícula–objeto (donde objeto es otra partícula o un grupo de partículas), pero el tiempo de cálculo es reducido significativamente respecto del cálculo directo de una partícula con todas. Por supuesto, se pleantea el problema de decidir cuando agrupar las partículas. El criterio más simple, introducido por Barnes y Hut, procede como sigue. Para cada partícula el cálculo de fuerza comienza en la raíz del árbol. El tamaño, s, de la celda a la que apunta una rama dada del árbol es comparado con la distancia d de la partícula al centro de masa de la celda. Si la relación

$$s/d < \theta$$

Se satisface, donde θ es un parámetro de tolerancia prefijado, entonces la estructura interna de la celda es ignorada y su contribución a la fuerza (a través de los términos monopolares y quadrupolares ya calculados) es agregada a la fuerza total sobre la partícula. Por el contrario, si el criterio no se satisface, esta celda es resuelta en sus descendientes, cada uno de los cuales es recursivamente examinado de acuerdo al criterio y, si es necesario, se vuelve a examinar más profundo en su extructura. Así, el proceso asciende a través del árbol, hasta que, ya sea el criterio se satisface o un hoja (esto es, una partícula) es alcanzada. En este último caso se computa la interacción partícula–partícula. Dado que el valor de θ determina cuan profundo se explora el árbol, lo podemos denominar *parámero de expansión* del árbol. Una mejora del criterio anterior, que implementamos en nuestro código, proviene del hecho de que criterio dado no distingue la distribución real de las partículas dentro de la celda, de modo que la contribución de una configuración donde una gran concentración de masa se localice en una de las esquinas será computada erroneamente. Para reducir este incoveniente, computamos también la distancia δ entre el centro geométrico y el centro de masa de la celda, y tomamos como criterio para no expandir la celda

$$s/\theta + \delta < d$$

En nuestra implementación es la subrutina,

[2] Dado que tratamos con fuerzas gravitatorias los momentos dipolares se anulan si se calcula relativos al centro de masa.

`calcularpotencial_aj` realiza el trabajo de computar la aceleración, en un dado instante, para todas las partículas. La misma llama en primer lugar a la subrutina `creararboloctal`, discutida en la sección anterior, para construir el árbol y efectúa el cálculo de la aceleración sobre cada partícula a través de la subrutina `recorrerarbolcalculophi` siguiendo las ideas presentadas.

Ejemplo. Con el fin de comprender, testear y verificar la construcción del árbol con el código implementado, construimos un programa principal, `main.f`, que llame a la subrutina `calcularpotencial_aj` para calcular el árbol para un sistema bidimensional de 10 partículas, cuya estructura de árbol generada por el código se ilustra en la fig. 2

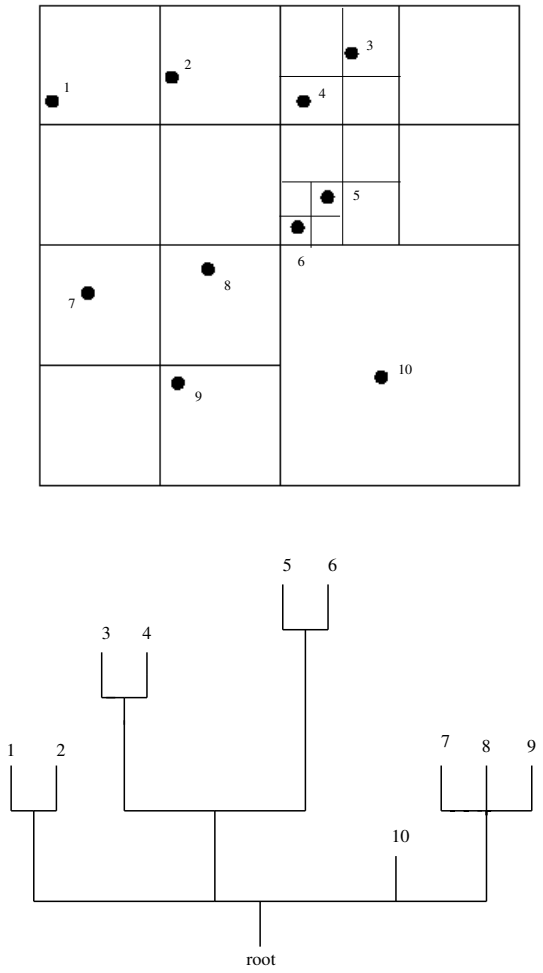

Figura 2. Relación entre la división espacial y la estructura de árbol.

Ejercicio 1. Descargar del sitio de la cátedra el archivo `arbol.tar.gz` que implementa el código de árbol octal. Compilar el mismo con la sentencia:

```
$ make
```

Ejercicio 2. Ejecutar el programa y estudiar la salida del mismo, la cual da la estructura del árbol, el desarrollo multipolar del mismo y las interacciones de las partículas.

Ejercicio 3. Verificar que los momentos cuadrupolares de las celdas calculados por el código son correctos.

Ejercicio 4. Experimentar con otros valores del parámetro de expansión del árbol para ver cómo éste influye en el cómputo de las iteracciones.