Particle Methods I + II

Stefan Lang

Interdisciplinary Center for Scientific Computing (IWR)
University of Heidelberg
INF 368, Room 532
D-69120 Heidelberg
phone: 06221/54-8264
email: Stefan.Lang@iwr.uni-heidelberg.de

WS 14/15

Topics

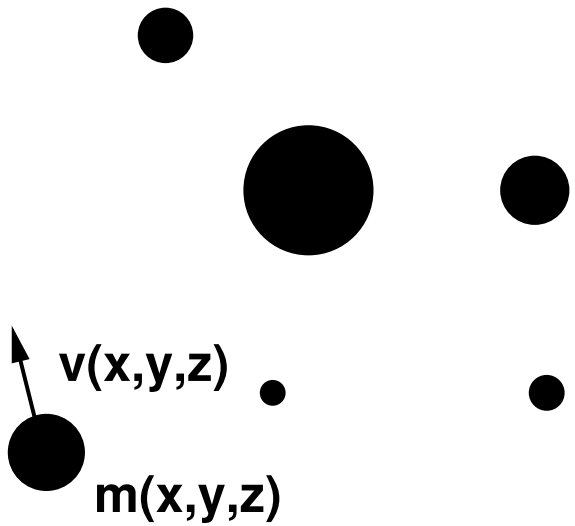
Particle methods

- Problem formulation
- Standard method
- Parallelisation
- Improvement of the method

Particle Methods

Task formulation:

- With particle methods one simulates the movement of N particles (or bodies) that move under the influence of a force field.
- The force field itself depends again on the position of the particles.
- The particles are characterised by point masses $m(x, y, z)$ and move with velocity $v(x, y, z)$ in the system.

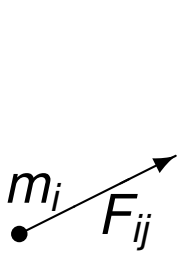


The N-Body Problem I

N-Body problem:

- Given being N point shaped masses $m_1, \ldots, m_N$ at positions $x_1, \ldots, x_N \in \mathcal{R}^3$.
- The gravitational force, excerted from body j onto body i, is given by Newton's law of gravitation

$$F_{ij} = \underbrace{\frac{\gamma\, m_i m_j}{\|x_j - x_i\|^2}}_{\text{strength}} \cdot \underbrace{\frac{x_j - x_i}{\|x_j - x_i\|}}_{\text{unit vector of } x_i \text{ in direction } x_j} \tag{1}$$



- The gravitational force can be written as potential gradient:

$$F_{ij} = m_i \frac{\gamma\, m_j (x_j - x_i)}{\|x_j - x_i\|^3} = m_i \nabla \left(\frac{\gamma\, m_j}{\|x_j - x_i\|} \right) = m_i \nabla \phi_j(x_i). \tag{2}$$

$\phi_y(x) = \frac{\gamma\, m}{\|y - x\|}$ is denoted gravitational potential of the mass m at position $y \in \mathcal{R}^3$.

The N-Body Problem II

- The movement equations of the considered N bodies are a consequence of the law force=mass$\times$acceleration:

$$\text{für } i = 1, \ldots, N$$

$$m_i \frac{dv_i}{dt} = m_i \nabla \left(\sum_{j \neq i} \frac{\gamma \, m_j}{\|x_j - x_i\|} \right) \tag{3}$$

$$\frac{dx_i}{dt} = v_i \tag{4}$$

  Here is $v_i(t) \colon \mathcal{R} \to \mathcal{R}^3$ the velocity of body i and $x_i(t) \colon \mathcal{R} \to \mathcal{R}^3$ the position in dependence of time.

- We get thus a system of ordinary differential equations of dimension $6N$ (three space dimensions). For its solution are still initial conditions for position and velocity required:

$$x_i(0) = x_i^0, \quad v_i(0) = v_i^0, \quad \text{for } i = 1, \ldots, N \tag{5}$$

Numerical Computation

- The integration of the movement equations (3) and (4) is performed numerically, since only for $N = 2$ analytical solutions are possible (Kegelcuts, Kepler's laws).
- The simplest method is the explicit Euler method:

$$\text{Discretisation in time:} \quad \begin{aligned} t^k &= k \cdot \Delta t, \\ v_i^k &= v_i(t^k), \\ x_i^k &= x_i(t^k). \end{aligned}$$

$$\left.\begin{aligned} v_i^{k+1} &= v_i^k + \Delta t \cdot \nabla\left(\sum_{j \neq i} \frac{\gamma\, m_j}{\| \underbrace{x_j^k - x_i^k}_{\text{„explicit“}} \|} \right) \\ x_i^{k+1} &= x_i^k + \Delta t \cdot v_i^k \end{aligned}\right\} \text{ for } i = 1, \ldots, N \qquad (6)$$

Problematique of Force Evaluation

- Surely there are better methods than the explicit Euler method, that only has a convergence order of $O(\Delta t)$, for the parallelisation this does not have a large impact, since the structure of other schemes is similar.
- The problem of force evaluation:
  In the force calculation the force of a body i depends on the position of *all* other bodies $j \neq i$. The effort for a force evaluation (that is at least once necessary for each time step) increases such as $O(N^2)$. In practical applications N can be in the range $10^6, \ldots, 10^9$, which means an enourmous computing effort.
- The main point of this chapter is therefore the presentation of improved sequential algorithms for fast force evaluation. These calculate the forces approximately with an effort of $O(N \log N)$ (There are methods with a complexity of $O(N)$ too, that we leave away for time reasons).

Parallelisation of the Standard Method

- The $O(N^2)$ algorithm is quite simple to parallelise. Each of the P processors gets $\frac{N}{P}$ bodies. To calculate the forces for all its bodies, the processor needs all other bodies. Herefore the data is shifted cyclic once through a ring topology.
- Analyse:

$$
\begin{aligned}
T_S(N) &= c_1 N^2 \\
T_P(N,P) &= c_1 P\Big(\underbrace{\frac{N}{P} \cdot \frac{N}{P}}_{\substack{\text{block } p \\ \text{with } q}} \Big) + \underbrace{c_2 P \frac{N}{P}}_{\text{communication}} = \\
&= c_1 \frac{N^2}{P} + c_2 N \\
E(P,N) &= \frac{c_1 N^2}{\left(c_1 \frac{N^2}{P} + c_2 N\right) P} = \frac{1}{1 + \frac{c_2}{c_1} \cdot \frac{P}{N}}
\end{aligned}
$$

constant efficiency for $N = O(P)$.

- Therefore is the iso-efficiency function because of $W = c_1 N^2$

$$W(P) = O(P^2)$$

(of course in relation to the sub-optimal standard algorithm).

Fast Multipole Methods

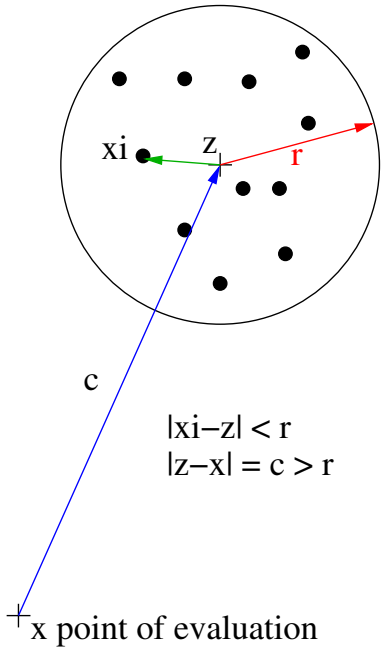
Fast Multipole Methods:

- The first basic idea has been published by *Pincus und Scherega* in 1977. Essential idea was the representation of a group of particles by a abstraction called pseudo particle. This represents the group properties and thus the resulting potential. The relationship with another particle group can then be calculated with a single multipole expansion.
- A second concept is the hierarchical coarsening of space in separate sub-areas.
- Both methods have been combined by Appel, 1985 and Barnes and Hut, 1986 within a single algorithm. The effort has a complexity of $O(NlogN)$.
- The fast multipole method has been published in 1987 by Greengard and Rokhlin. To both mentioned ideas they further introduce a local expansion of potentials. In specific cases, e.g. for uniform particle distribution, the effort reduces to $O(N)$.

Fast Summation Methods I

Accelerated method for force evaluation:

- We consider the figured cluster of bodies: M mass points be contained in a circle around z with radius r. We evaluate the potential ϕ of all mass points in point x with $\|z - x\| = c > r$.



- Let us consider first a mass point at position ξ with $\|\xi - z\| < r$. The potential of mass in ξ be (the multiplicative factor γm is neglected)

$$\phi_\xi(x) = \frac{1}{\|\xi - x\|} = f(\xi - x).$$

Fast Summation Methods II

- The potential depends only on the distance $\xi - x$.
- Now we insert the point z and develop in a taylor series around $(z - x)$ up to order p (do not interchange with processors):

$$\begin{aligned} f(\xi - x) &= f((z - x) + (\xi - z)) = \\ &= \sum_{|\alpha| \leq p} \frac{D^\alpha f(z - x)}{|\alpha|!} (\xi - z)^{|\alpha|} + \underbrace{\sum_{|\alpha| = p} \frac{D^\alpha f(z - x + \theta(\xi - z))}{|\alpha|!} (\xi - z)^{|\alpha|}}_{\text{remainder term}} \end{aligned}$$

  for a $\theta \in [0, 1]$. Important is the separation of variables x and ξ.
- The size of the error (remainder term) depends on p, r and c.
- How can the series expansion be used to accelerate the potential evaluation?
- Herefore we assume that the evaluation of the potential of M masses is to be computed at N points, which normally would require $O(MN)$ operations.

Fast Summation Methods III

- For the evaluation of the potential at the position x we calculate

$$\begin{aligned}
\phi(x) &= \sum_{i=1}^{M} \gamma\, m_i \phi_{\xi_i}(x) = \sum_{i=1}^{M} \gamma\, m_i f((z-x)+(\xi_i - z)) \approx \\
&\underset{\substack{\text{(taylor series up}\\ \text{to order } p)}}{\approx} \sum_{i=1}^{M} \gamma\, m_i \sum_{|\alpha|\le p} \frac{D^\alpha f(z-x)}{|\alpha|!}(\xi_i - z)^{|\alpha|} = \\
&\underset{\text{(permute sum)}}{=} \sum_{|\alpha|\le p} \frac{D^\alpha f(z-x)}{|\alpha|!} \Bigg(\underbrace{\sum_{i=1}^{M} \gamma\, m_i (\xi_i - z)^{|\alpha|}}_{\substack{=:\, M_\alpha,\\ \text{independent of } x!}} \Bigg) = \\
&= \sum_{|\alpha|\le p} \frac{D^\alpha f(z-x)}{|\alpha|!} M_\alpha
\end{aligned}$$

- The calculation of coefficients M_α requires once $O(Mp^3)$ operations.
- Are the M_α known, then a evaluation of $\phi(x)$ costs $O(p^5)$ operations.
- For evaluation at N points we get such the total complexity of $O(Mp^3 + Np^5)$.

Fast Summation Methods IV

- It is clear that the potential, calculated in this way, is not exact. The algorithm only makes sense, if the error can be controlled such that it is neglectable (e.g. smaller as the discretization error).
- A criteria for error control provides the error estimation:

$$\frac{\phi_\xi(x) - \sum_{|\alpha|\leq p} \frac{D^\alpha f(z-x)}{|\alpha|!}(\xi - z)^{|\alpha|}}{\phi_\xi(x)} \leq c(2h)^{p+1},$$

  with $\frac{r}{c} \leq h < \frac{1}{4}$. For the case $c > 4r$ the error reduces by $(1/2)^{p+1}$.
- The approximation is then the more accurate
  - the smaller $\frac{r}{c}$
  - the larger the degree of expansion p.

Gradient Calculation

- In the N-body problem one does not want to calculate the potential, but the force, thus the gradient of the potential.
- This works via

$$\frac{\partial \phi(x)}{\partial x_{(j)}} \overset{\text{series dev.}}{\approx} \frac{\partial}{\partial x_{(j)}} \sum_{|\alpha| \leq p} \frac{D^\alpha f(z-x)}{|\alpha|!} M_\alpha = \sum_{|\alpha| \leq p} \frac{D^\alpha \partial_{x_{(j)}} f(z-x)}{|\alpha|!} M_\alpha$$

(j) ↑ space dimension

- One has thus only to calculate $D^\alpha \partial_{x_{(j)}} f(z-x)$ instead $D^\alpha f(z-x)$.
- Above we have used a Taylor series. This is not the only possibility of a series expansion. Besides there are for the considered potentials $\frac{1}{\log(\|\xi - x\|)}$ (2D) and $\frac{1}{\|\xi - x\|}$ (3D) other expansions, the so called multipole expansions, that are fitting better.
- For this series a better error estimation applies in the sense, that they have the form

$$error \leq \frac{1}{1 - \frac{r}{c}} \left(\frac{r}{c}\right)^{p+1}$$

and therefore are already for $c > r$ satiesfied.
- Moreover the complexity in relation to p is better (p^2 in 2D, p^4 in 3D).

Gradient Calculation

- An approximation of the gravitation potential, that is often used by physicans, is a taylor expansion of

$$\phi(x) = \sum_{i=1}^{M} \frac{\gamma\, m_i}{\|(s-x) + (\xi_i - s)\|},$$

  where s is the *point of gravity* of M masses (and not a fictitious circle midpoint).
- The so-called monopole expansion reads

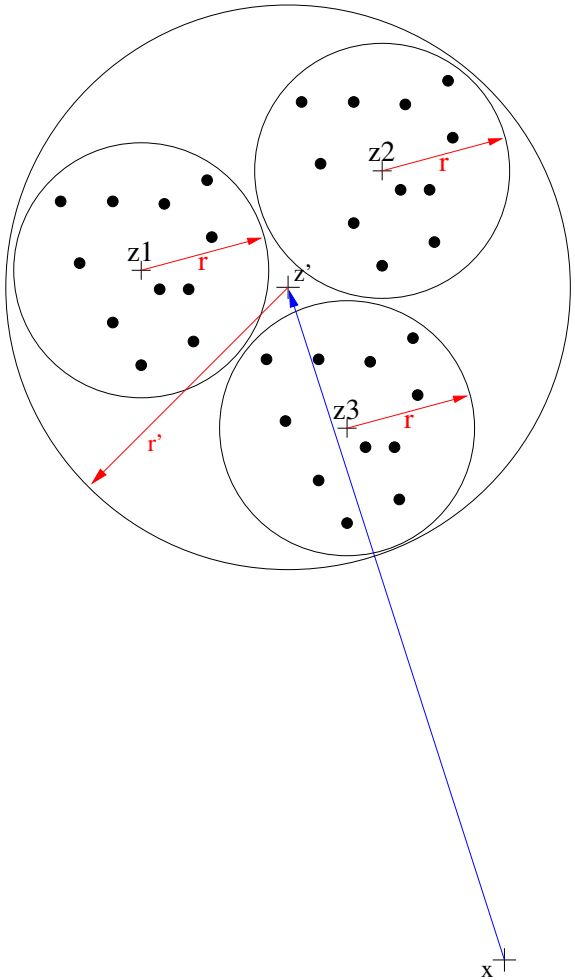
$$\phi(x) \approx \frac{\sum_{i=1}^{M} \gamma\, m_i}{\|s-x\|}$$

  (this means a body of mass $\sum m_i$ in s).
- The accuracy is then only controlled by the relationship r/c.

Shifting of an Expansion

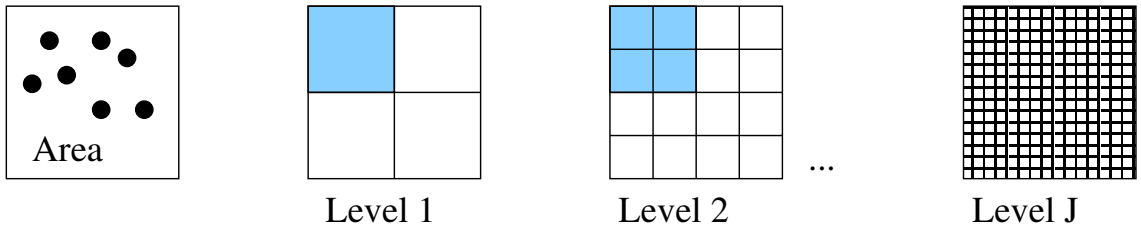
- In the following algorithms we still need a further tool, that concerns the shifting of expansions.
- The mapping showns three clusters of bodies in circles around z_1, z_2, z_3 each with radius r. The three circles are contained in a larger circle around z' with radius r'.
- If we want to evaluate the potential of all masses in x with $\|x - z'\| > r'$, we could use the series expansion around z'.
- If already series expansions have been calculated in the three smaller circles (this means the coefficients M_α), then the expansion coefficients of the new series can be computed from the one of the old series with an effort of $O(p^\alpha)$, thus independent of the number of masses.
- Here *no* additional error arises, and it applies also the error estimation with appropriate larger r'.

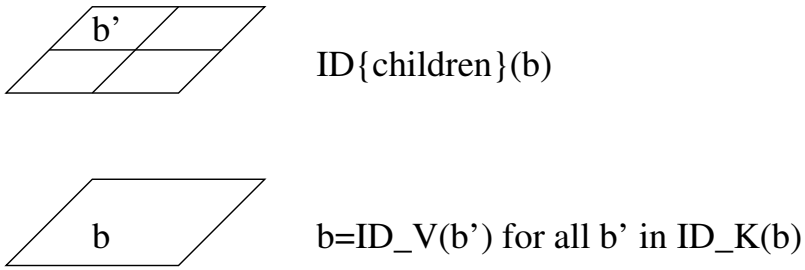
Shifting of an Development



Uniform Point Distribution

- First we develop an algorithm, that is usable for a uniform point distribution. This has the advantage of simpler data structures and the possibility of simpler load balancing. We present the ideas for the two-dimensional case, since this can be drawn easier. However, all can be performed analogous for three space dimensions.
- All bodies be contained in a square $\Omega = (0, D_{max})^2$ of side length D_{max}. We now introduce for Ω a hierachy of steadily finer grids. Level l is developed from level $l-1$ by quatering of elements.



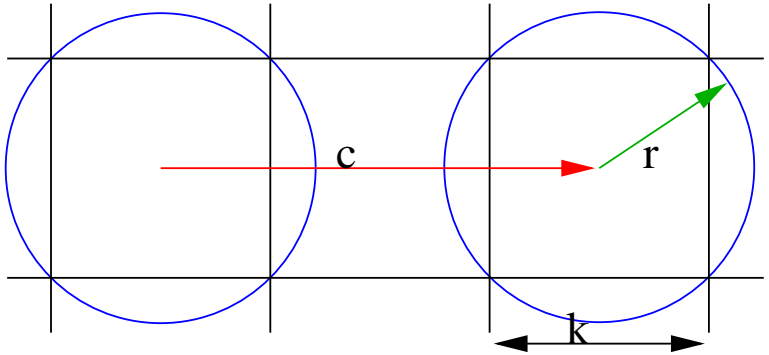


construction of grids

Uniform Point Distribution

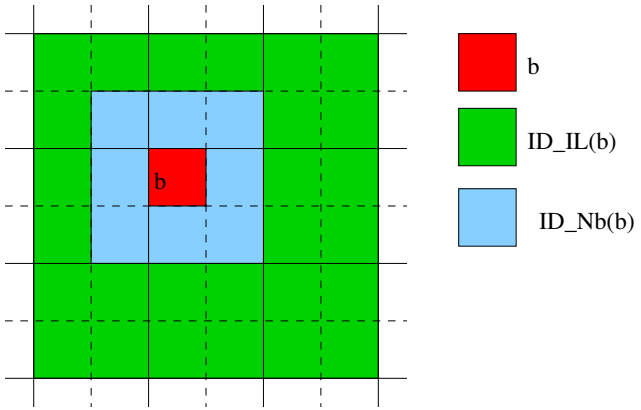
- If we want s bodies per fine grid box, then applies $J = \log_4\left(\frac{N}{s}\right)$. For two not-neighbored squares we get the following estimation for the r/c relationship (masses in b, evaluation in a)

$$\begin{aligned} r &= \sqrt{2}\frac{k}{2} \\ c &= 2k \\ \Rightarrow \frac{r}{c} &= \frac{\sqrt{2}\,k}{4k} = \frac{\sqrt{2}}{4} \approx 0.35. \end{aligned}$$



r/c evaluation for two not-neighbored squares

- For an element b on level l one defines the following regions in the neighborhood of b:
  - $Nb(b) =$ alle *neigbors* b' of b on level l ($\partial b \cap \partial b' \neq \emptyset$).
  - $IL(b) =$ *interaction list* of b: children of neighbors of *father*(b), that are not neighbors of b.



Uniform Point Distribution

The following algorithm calculates the potential at all positions $x_1, \ldots, x_N$:

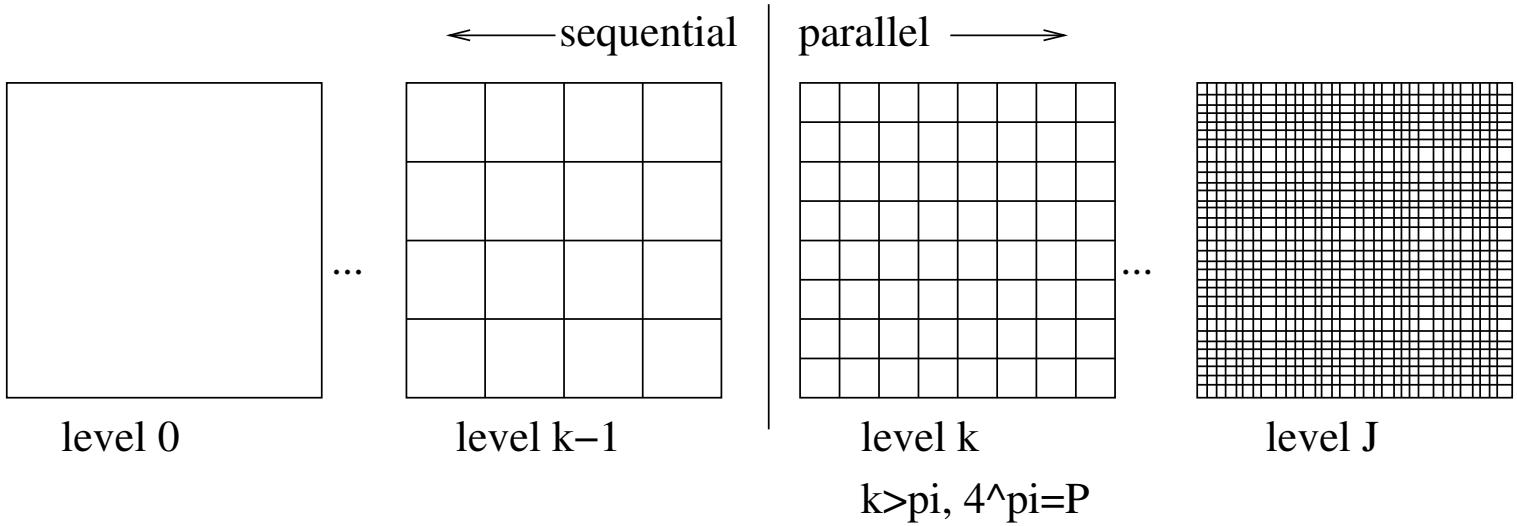
	effort
1. Preparation phase:	
For each fine grid box calculate a far field expansion;	$O(\frac{N}{s} sp^{\alpha})$
For all levels $l = J-1, \ldots, 0$	
For each box b on level l	
calculate expansion in b from expansion in $children(b)$;	$O(\frac{N}{s} sp^{\gamma})$
2. Evaluation phase:	
For each fine grid box b	
For each body q in b	
{	
calculate exact potential of all $q' \in B, q' \neq q$;	$O(Ns)$
For all $b' \in Nb(b)$	
For all $q' \in b'$	
calculate potential of q' in q;	$O(N8s)$
$\bar{b} = b$;	
For all levels $l = J, \ldots, 0$	
{	
For all $b' \in IL(\bar{b})$	
Evaluate the far field expansion of b' in q;	$O(\frac{N}{s} sp^{\beta})$
}	
}	

Uniform Point Distribution

- Total effort: $O(N \log N p^\gamma + Ns + Np^\alpha + \frac{N}{s} p^\beta)$, thus asymptotically $O(N \log N)$.
- Here denotes α the exponents for the building of the far field expansion and β the exponents for shifting.
- One considers, that one has N/s bodies per box on level J because of the uniform distribution.
- The accuracy is controlled here by the expansion degree p, the relationship r/c is fixed.

Parallelisation: Load Balancing

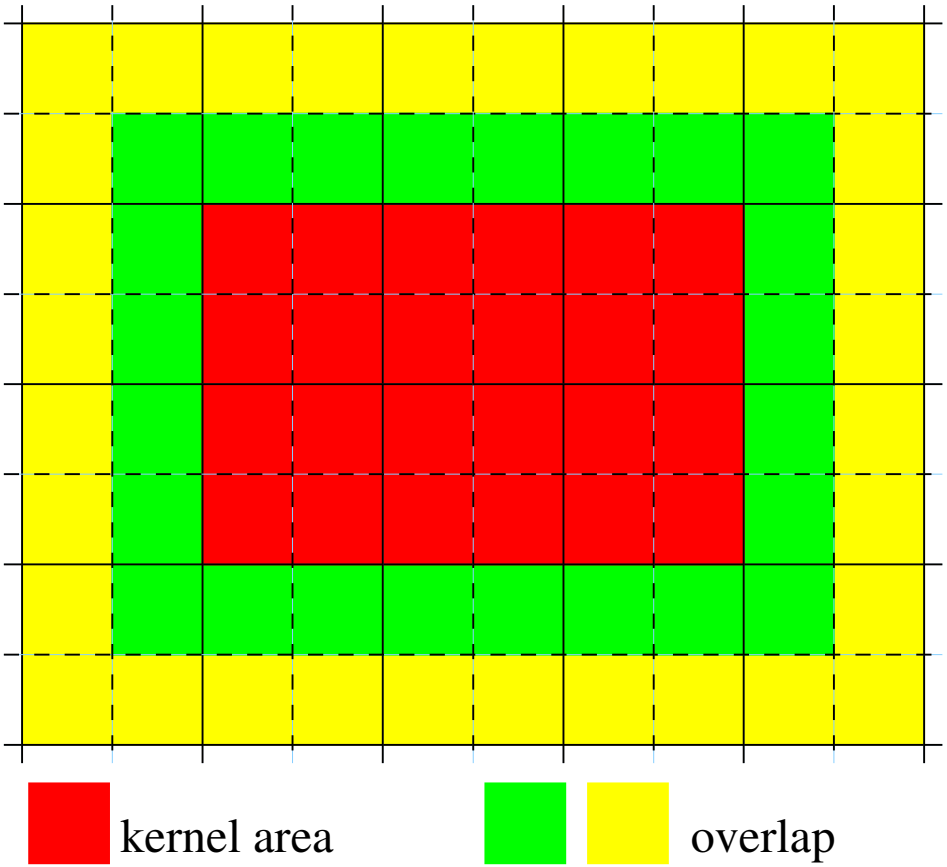
- Load balancing:
  The grid with the associated bodies is distributed onto the processors.
- Since we now have a hierarchy of grids we progress as follows:
  - Each processor shall at least get 2×2 grid cells.
  - Be $P = 4^{\pi}$, then choose $k = \pi + 1$ and distribute the grid on level k onto all processors (each has 2×2 elements).
  - All levels $l < k$ are stored on all processors.
  - All levels $l > k$ are distributed such, that $children(b)$ are processed in the same processor than b.
  - Example for $P = 16 = 4^2$.



Distribution of boxes for the parallelisation

Parallelisation: Overlap

- Additional to the assigned elements each processor stores further an overlap area of two element rows:



overlap area of a processor

Parallelisation: Analysis I

- Since each has at least 2×2 elements, the overlap area lies always in directly neighbored processors.
- The evaluation of *IL* requires at most communication with nearest neighbors.
- To establish the far field expansion for the levels $l < k$ is a all-to-all communication required.
- Scalability estimation: Be $\frac{N}{sP} \gg 1$. Because of

$$J = \log_4\left(\frac{N}{s}\right) = \log_4\left(\frac{N}{sP}P\right) = \underbrace{\log_4\left(\frac{N}{sP}\right)}_{J_p} + \underbrace{\log_4 P}_{J_s}$$

J_s levels are computed sequentially and $J_p = O(1)$ levels in parallel.

Parallelisation: Analysis II

- Then we get for *fixed expansion degree:*

$$T_P(N,P) \overset{(\frac{N}{P}=const.)}{=} \underbrace{c_1 \frac{N}{P}}_{\substack{\text{FFE level } J \ldots K \\ \text{evaluate near field}}} + \underbrace{c_2 \operatorname{ld} P + c_3 P}_{\substack{\text{all-to-all. This is always} \\ \text{data for four cells}}} + \underbrace{c_4 p}_{\substack{\text{compute FFE in whole } \Omega \\ \text{for } l = k-1 \ldots 0 \text{ in } \textit{all} \\ \textit{processors}}} + \underbrace{c_5 J_p \frac{N}{P}}_{\text{FFE levels } l \geq k} + \underbrace{c_5 \frac{N}{P} J_s}_{\text{FFE levels } l < k}$$

- Thus:

$$E(N,P) = \frac{c_5 N \log N}{\Big(c_5 \frac{N}{P}\underbrace{(J_s + J_p)}_{\log N} + \underbrace{(c_3 + c_4)P}_{\substack{\text{all to all coarse} \\ \text{grid FFE}}} + \underbrace{c_2 \operatorname{ld} P}_{\text{all-to-all}} + \underbrace{c_1 \frac{N}{P}}_{\substack{\text{nearfield local FFE}}}\Big) P} =$$

$$= \frac{1}{1 + \frac{c_3+c_4}{c_5} \cdot \frac{P^2}{N \log N} + \frac{c_2}{c_5} \cdot \frac{P \operatorname{ld} P}{N \log N} + \frac{c_1}{c_5} \cdot \frac{1}{\log N}}$$

- For an iso-efficient scaling N has thus to grow nearly like P^2!

Parallelisation: Irregular Distribution

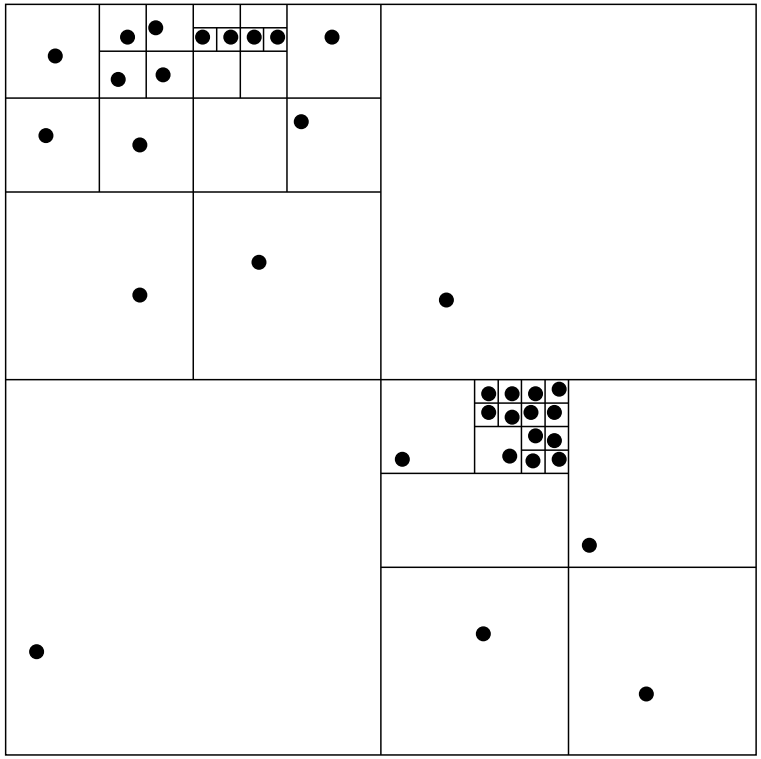
- The assumption of a uniform distribution of bodies is in some application (e.g. astro physics) not realistic.
- If we want to have in each fine grid box exactly a single body and is D_{min} the minimal distance between two bodies, then one needs a grid with

$$\log L = \log \frac{D_{max}}{D_{min}}$$

  grid levels. L is called *„separation ratio“.*
- But from these the most are empty. As with sparse matrices one now avoids to store the empty cells. In two space dimensions this construction is called „adaptive quadtree“.
- The adaptive quadtree is constructed in the following way:
  - Initialisation: root contains all bodies in the square $(0, D_{max})$.
  - As long as a leaf b with more than two bodies exists:
    - Subdivide b into four parts
    - Put each body of b into the appropriate child
    - Leave out empty children.

Parallelisation: Irregular Distribution

- Example of an adaptive quadtree:



- The effort amounts (sequentially) to $O(N \log L)$.
- The first (successful) fast evaluation algorithm for irregular distributed bodies has been proposed by Barnes and Hut
- As in the uniform case the far field expansion is constructed from the leafs up to the root (at Barnes & Hut: monopole expansion).

Irregular Distribution

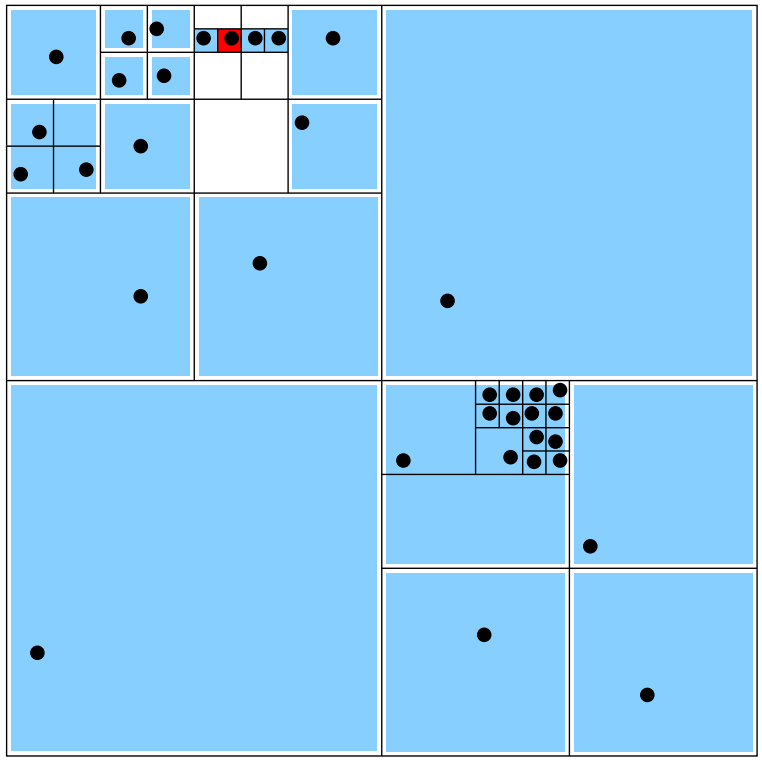
- For a body q one can then calculate the potential in q with the following recursive algorithm:

```
Pot(body q, box b)
{
    double pot = 0;
    if (b is leaf ∧ q = b.q) return 0;          // end
    if (children(b) == ∅)
        return φ(b.q, q);                       // direct evaluation
    if (r(b) / dist(q,b) ≤ h)
        return φ_b(q);                          // evaluate FFE
    for (b' ∈ children(b))
        pot = pot + Pot(q, b');                 // recursive descent
    return pot;
}
```

- For the calculaton *Pot* is called with q and the root of the quadtree.
- In the algorithm of Barnes & Hut the accuracy of the evaluation is controlled with the parameters h.

Irregular Distribution

Which cells of the quadtree are visited in the Barnes & Hut algorithm?



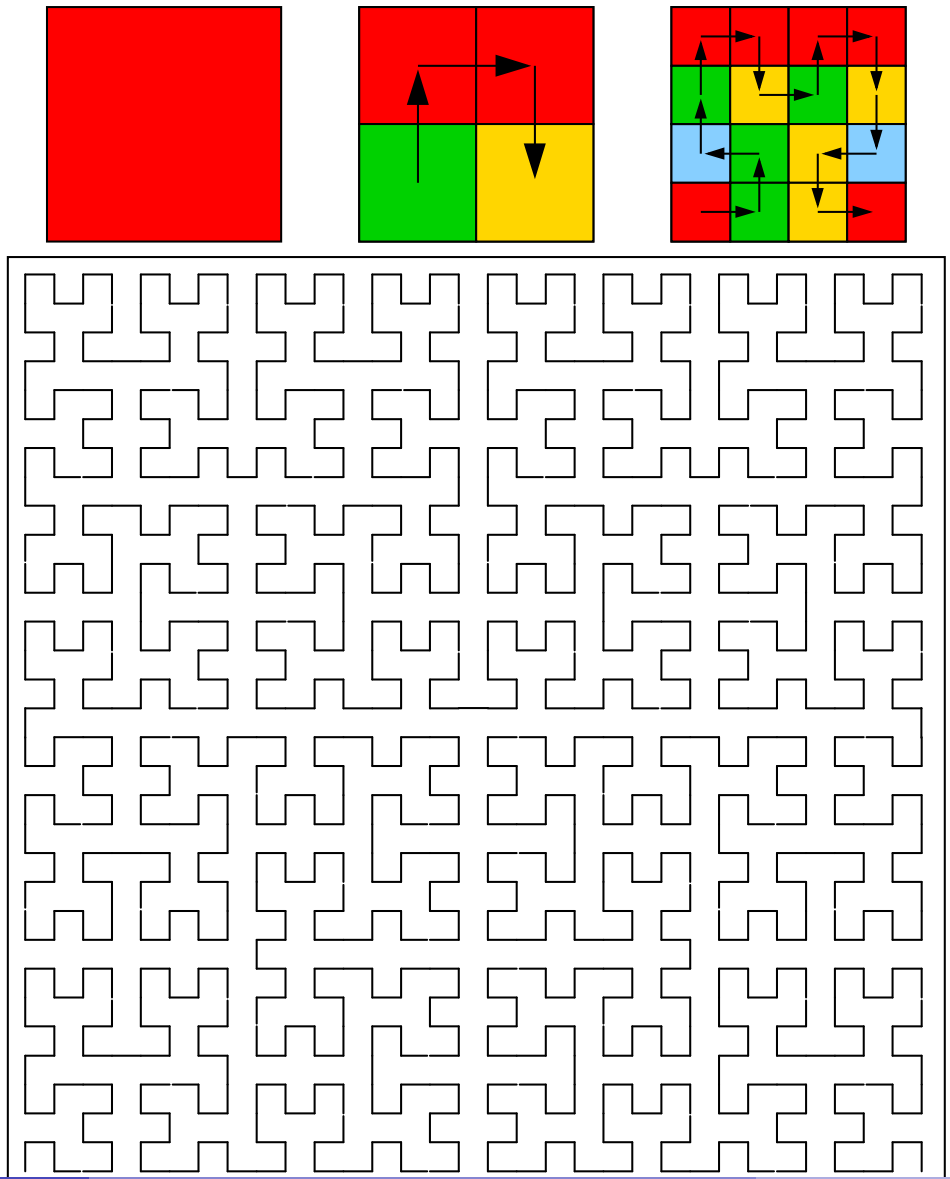
Example to the evaluation in the Barnes & Hut algorithm

Irregular Distribution: Parallelisation I

- The parallelisation of this algorithm is relatively complex, such that we can only provide some hints. For details we point out Salmon, 1994.
- Since the positions of the particle change with time, the adaptive quadtree has to be constructed in each time step. Furthermore the partitioning of the bodies onto the processors has to be done in such a way, that close neighboring bodies are also stored in close neighboring processors.

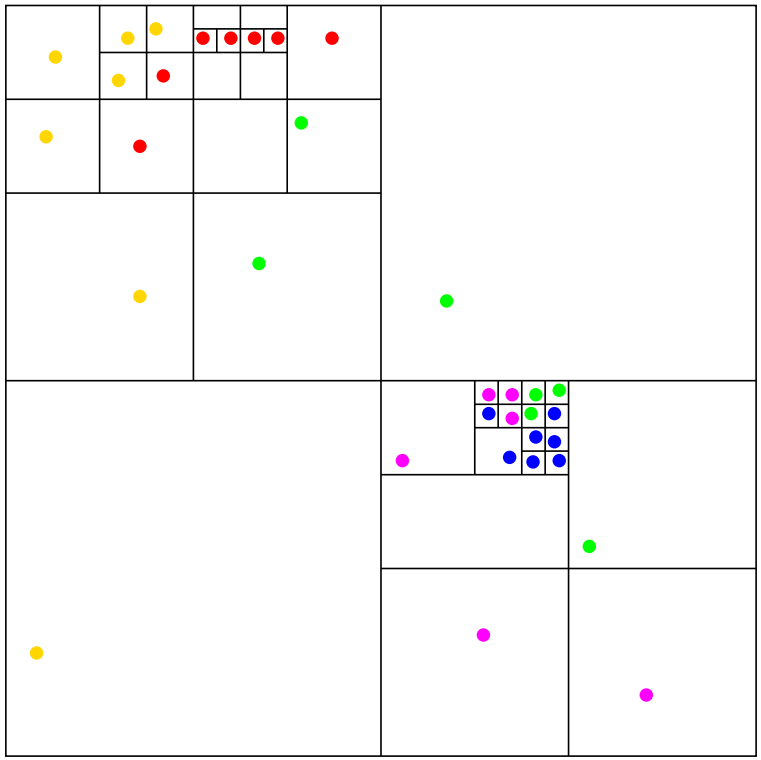
Irregular Distribution: Parallelisation II

- A very fancy load balancing method works with „space filling curves“. The so called Peano or Hilbert curves have the following shape:



Irregular Distribution: Parallelisation III

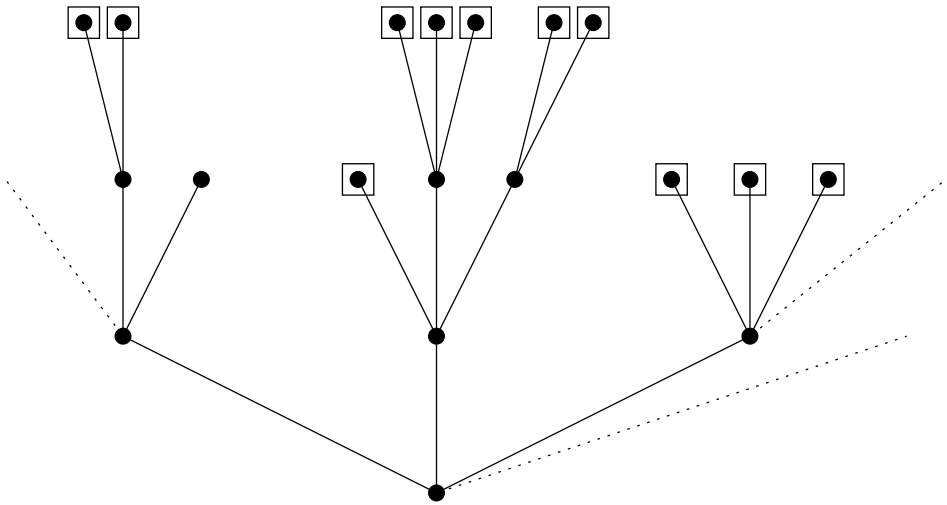
- A Hilbert curve of appropriate depth can be used to find a linear ordering of the bodies (resp. the leafs of in the quad tree). This can then easily be partitioned into P sections of length N/P.



- Salmon & Warren show that with this data distribution the adaptive quadtree can be constructed in parallel with few communication. Similar to the uniform algorithm the coarse grid information, that all processors store, can then be constructed by an all-to-all communication.

Parallel Construction of the Adaptive Quadtree

- *Starting point:* Each processor has a set of bodies, that corresponds to *a single* connected section on the Hilbert curve.
- *Step 1:* Each processor constructs locally the quadtree for *its* bodies. The „numbers“ of the leafs are then increasing.

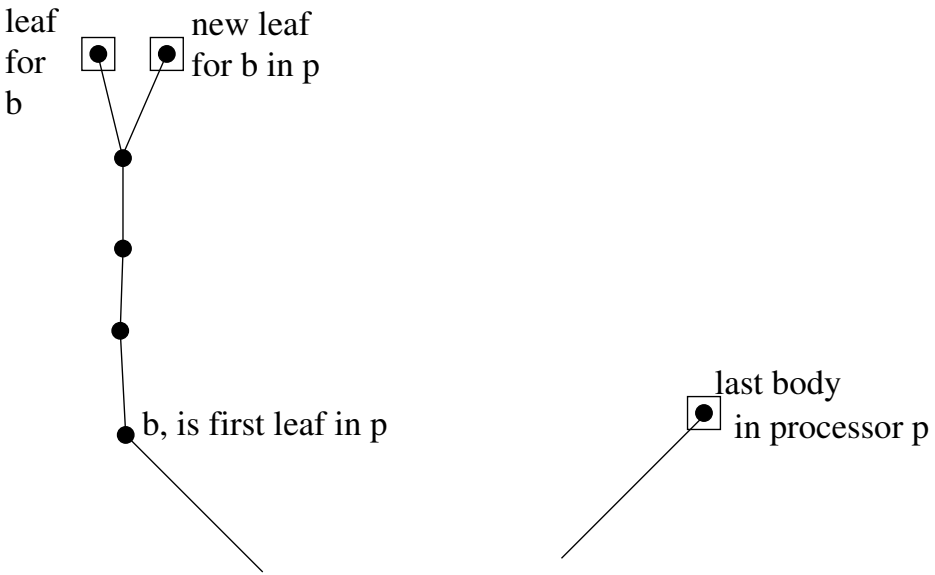


Local quadtree

- *Step 2:* Comparison with neighboring processors. Question: Would a sequential program create for the bodies of processor p the same leafs? In general no, since a body of p and a body of $q \neq p$ could share a box.

Parallel Construction of the Adaptive Quadtree

- What can happen?
  Be b the *first body* in processor p and b' the *last* in processor $p-1$. Now there are two possibilities in processor p:
  1. Body b' lies in the same box as body b. $\Longrightarrow$ Subdivide the box until both bodies are separated. The new leaf is the same, that also a sequential program would have calculated! If this *wouldn't* be such, then there had to exist a body b'' in Prozessor $q \neq p$ such that $b'' \in$ new leaf of b. But this is impossible, since all b'' are ordered before b' or after the last node of processor p!



Exchange of the boundary leafs

  2. Body b' lies not in the same box as body b. $\Longrightarrow$ there is nothing to do.
- For the last body in p and the first in $p+1$ we do the same!

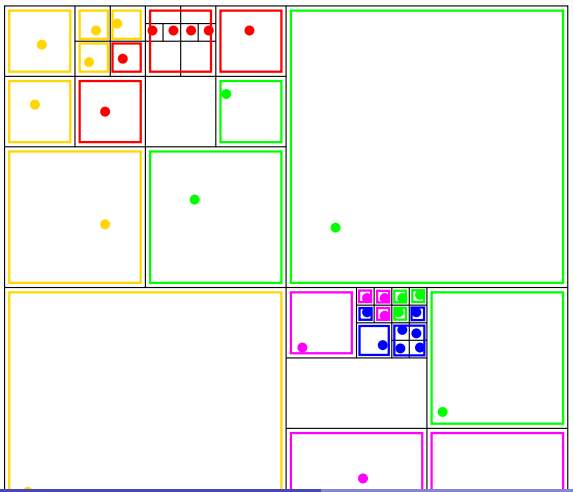
The Coarse Grid Problem

- Like in the uniform case the quadtree is stored from root up to a certain depth in each processor, such that for these far field evaluations no communication is necessary or, if needed, the processor is known that stores the appropriate information.

Definition

A box *b* in the quadtree is called *branch node*, if *b* has only bodies of processor *p*, but the father of *b* however contains bodies of different processors. This processor *p* „belongs“ the branch node.

- All boxes of the quadtree from the root to the branch nodes inclusively are stored on all processors.



Force Evaluation & Communication

- If for evaluation a recursive call in a branch node is necessary, then the message is sent to the according processor, that is owner of the branch node. This answers the request asynchronously and sends back the result.
- After the update of positions one calculates a new Hilbert numbering (can be performed in $\frac{N}{P} \log L$ without quadtree construction) for the local bodies. Then each gets again a section of length $\frac{N}{P}$. This works e.g. with a parallel sorting algorithm!
- Salmon & Warren can simulate with their implementation 322 Millionen bodies (!) using 6800 processors (!, Intel ASCI-Red).