Simulation and High-Performance Computing

Part 10: Non-local force fields

Steffen Börm

Christian-Albrechts-Universität zu Kiel

October 2nd, 2020

Example: Gravitation

Task: Given n planets at positions $y_1, \ldots, y_n \in \mathbb{R}^2$ with masses $m_1, \ldots, m_n \in \mathbb{R}$, we want to evaluate the gravitational potential in $x \in \mathbb{R}^2$, given by

$$\varphi(x) = \gamma \sum_{j=1}^{n} m_j \frac{1}{\|y_j - x\|}.$$

Example: Gravitation

Task: Given n planets at positions $y_1, \ldots, y_n \in \mathbb{R}^2$ with masses $m_1, \ldots, m_n \in \mathbb{R}$, we want to evaluate the gravitational potential in $x \in \mathbb{R}^2$, given by

$$\varphi(x) = \gamma \sum_{j=1}^{n} m_j \frac{1}{\|y_j - x\|}.$$

Reformulation: Describe non-local interactions by the kernel function

$$g(x, y) = \frac{\gamma}{\|y - x\|}, \qquad \varphi(x) = \sum_{j=1}^{n} g(x, y_j)\, m_j.$$

Example: Gravitation

Task: Given n planets at positions $y_1, \ldots, y_n \in \mathbb{R}^2$ with masses $m_1, \ldots, m_n \in \mathbb{R}$, we want to evaluate the gravitational potential in $x \in \mathbb{R}^2$, given by

$$\varphi(x) = \gamma \sum_{j=1}^{n} m_j \frac{1}{\|y_j - x\|}.$$

Reformulation: Describe non-local interactions by the kernel function

$$g(x, y) = \frac{\gamma}{\|y - x\|}, \qquad \varphi(x) = \sum_{j=1}^{n} g(x, y_j)\, m_j.$$

Challenge: Evaluating $\varphi(x)$ requires at least n operations.
If we have to evaluate in many points, this will take a long time.

Low-rank approximation

Low-rank approximation: If we can find $a_1, \ldots, a_k$ and $b_1, \ldots, b_k$ with

$$g(x,y) \approx \sum_{\nu=1}^{k} a_\nu(x)\, b_\nu(y),$$

we can approximate the potential by

$$\begin{aligned}
\varphi(x) &= \sum_{j=1}^{n} g(x,y_j)\, m_j \approx \sum_{j=1}^{n} \sum_{\nu=1}^{k} a_\nu(x)\, b_\nu(y_j)\, m_j \\
&= \sum_{\nu=1}^{k} a_\nu(x) \sum_{j=1}^{n} b_\nu(y_j)\, m_j = \sum_{\nu=1}^{k} a_\nu(x)\, z_\nu, \qquad z_\nu := \sum_{j=1}^{n} b_\nu(y_j)\, m_j.
\end{aligned}$$

Approach: If we have prepared $z_1, \ldots, z_k$ in advance, approximating $\varphi(x)$ requires $\sim k$ operations. If $k \ll n$, this can be very efficient.

Tensor interpolation

Goal: Find low-rank approximation of g.

Approach: Interpolate first in the y_1 variable, then in y_2.

$$\begin{aligned} g(x,y) &\approx \sum_{\nu_1=0}^{m} g(x,(\xi_{1,\nu_1},y_2))\,\ell_{1,\nu_1}(y_1) \\ &\approx \sum_{\nu_1=0}^{m}\sum_{\nu_2=0}^{m} g(x,(\xi_{1,\nu_1},\xi_{2,\nu_2}))\,\ell_{1,\nu_1}(y_1)\,\ell_{2,\nu_2}(y_2) \\ &= \sum_{\nu\in M} g(x,\xi_\nu)\,\ell_\nu(y) \end{aligned}$$

with $M := [0:m]\times[0:m]$ and

$$\xi_\nu := (\xi_{1,\nu_1},\xi_{2,\nu_2}), \qquad \ell_\nu(y) := \ell_{1,\nu}(y_1)\,\ell_{2,\nu_2}(y_2) \qquad \text{for all } \nu\in M.$$

Fictitious planets

Interpolation yields

$$g(x,y) \approx \sum_{\nu\in M} g(x,\xi_\nu)\,\ell_\nu(y).$$

Gravitational potential approximated by

$$\varphi(x) = \sum_{j=1}^{n} g(x,y_j)\,m_j \approx \sum_{\nu\in M} g(x,\xi_\nu) \sum_{j=1}^{n} \ell_\nu(y_j)\,m_j$$

$$= \sum_{\nu\in M} g(x,\xi_\nu)\,z_\nu \qquad \text{with} \qquad z_\nu := \sum_{j=1}^{n} \ell_\nu(y_j)\,m_j.$$

Interpretation: Our n planets are replaced by “fictitious planets” at positions ξ_ν with masses z_ν.

Admissibility

Problem: Interpolation is only accurate if the function g is smooth enough to be approximated by a polynomial.

Admissibility

Problem: Interpolation is only accurate if the function g is smooth enough to be approximated by a polynomial.

Error analysis: If we interpolate g on a box $B = [a_1, b_1] \times [a_2, b_2]$, we can expect a good accuracy if

$$\mathrm{diam}(B) \leq \eta \; \mathrm{dist}(x, B)$$

holds with an admissibility parameter $\eta > 0$.

Admissibility

Problem: Interpolation is only accurate if the function g is smooth enough to be approximated by a polynomial.

Error analysis: If we interpolate g on a box $B = [a_1, b_1] \times [a_2, b_2]$, we can expect a good accuracy if

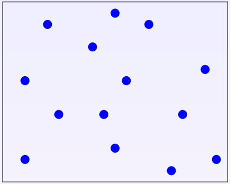
$$\operatorname{diam}(B) \leq \eta \operatorname{dist}(x, B)$$

holds with an admissibility parameter $\eta > 0$.

Problem: The box B has to be large enough to contain all y_j, so the admissibility condition will only be satisfied if x happens to be positioned far away from all planets.

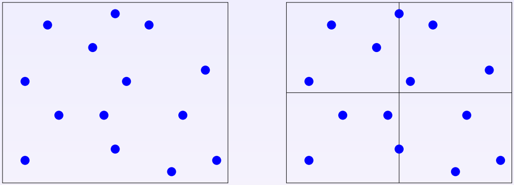
Clusters

Idea: If a box B is too large, split it into smaller boxes.



Clusters

Idea: If a box B is too large, split it into smaller boxes.

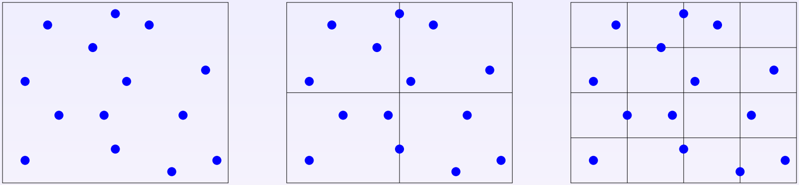


Cluster tree:

- Start with a box containing all planets.
  This box is the root of the cluster tree.
- If a box σ contains only a few planets, stop.
- Otherwise, split σ into smaller boxes σ'.
  These smaller boxes become the children of σ.

Clusters

Idea: If a box B is too large, split it into smaller boxes.



Cluster tree:

- Start with a box containing all planets.
  This box is the root of the cluster tree.
- If a box σ contains only a few planets, stop.
- Otherwise, split σ into smaller boxes σ'.
  These smaller boxes become the children of σ.

Index sets

Idea: For each cluster σ, define

$$\hat{\sigma} := \{j \in [1:n] \ : \ y_j \in \sigma\}.$$

Precise definition:

- $y_j \in \sigma$ for all $j \in \hat{\sigma}$.
- If σ is the root cluster, $\hat{\sigma} = [1:n]$.
- If σ has children, we have

$$\hat{\sigma} = \bigcup_{\sigma' \in \mathrm{chil}(\sigma)} \hat{\sigma}'.$$

- If σ_1, σ_2 are children of σ, we have

$$\sigma_1 \neq \sigma_2 \quad \Longrightarrow \quad \hat{\sigma}_1 \cap \hat{\sigma}_2 = \emptyset.$$

Recursion

Goal: Approximate the potential

$$\varphi(x) = \sum_{j=1}^{n} g(x, y_j)\, m_j.$$

Root cluster: If σ is the root cluster, we have $\hat\sigma = [1 : n]$ and therefore

$$\varphi(x) = \sum_{j\in\hat\sigma} g(x, y_j)\, m_j.$$

Recursion

Goal: Approximate the potential

$$\varphi(x) = \sum_{j=1}^{n} g(x, y_j)\, m_j.$$

Root cluster: If σ is the root cluster, we have $\hat{\sigma} = [1 : n]$ and therefore

$$\varphi(x) = \sum_{j \in \hat{\sigma}} g(x, y_j)\, m_j.$$

Recursion: If σ is not admissible with respect to x, use

$$\varphi(x) = \sum_{\sigma' \in \mathrm{chil}(\sigma)} \sum_{j \in \hat{\sigma}'} g(x, y_j)\, m_j$$

and compute the sums for the children σ' recursively.
Stop the recursion once the cluster is a leaf or admissible.

Implementation: Cluster

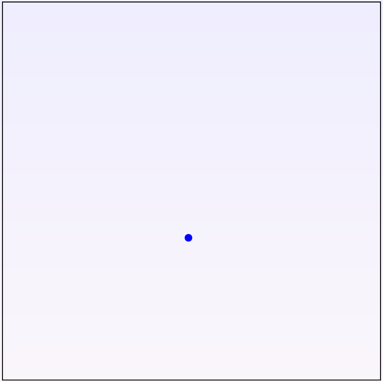
```
typedef struct cluster_struct cluster;

struct cluster_struct {
  /* Bounding box */
  real a[2], b[2];

  /* Interpolation points */
  real *xi_1d[2];
  real (*xi)[2];

  /* Children */
  int children;
  cluster **chil;
};
```

Children represented by `s->chil[i]` with $i \in [0 : \texttt{children} - 1]$.

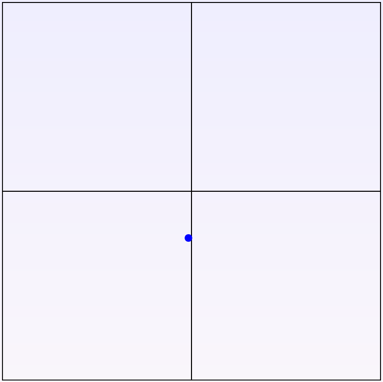
Implementation: Recursion

```
real
eval_cluster(real x[2], const cluster *s)
{
  if(diam(s) <= dist(x, s)) {
    for(nu=0; nu<k; nu++)
      result += potential(x, s->xi[nu]) * s->z[nu];
  }
  else if(s->children > 0) {
    for(i=0; i<s->children; i++)
      result += eval_cluster(x, s->chil[i]);
  }
  else {
    for(i=0; i<s->size; i++)
      result += potential(x, y[s->idx[i]]);
  }
  return result;
}
```

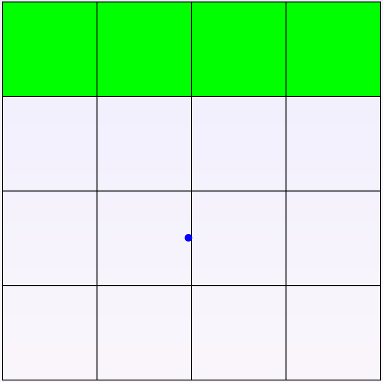
Example: Recursion



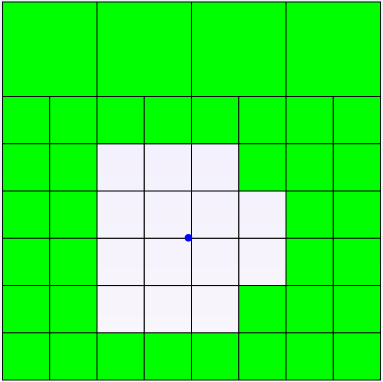
Example: Recursion



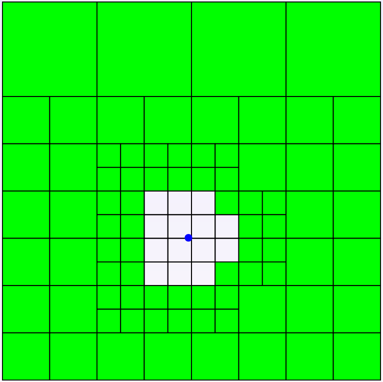
Example: Recursion



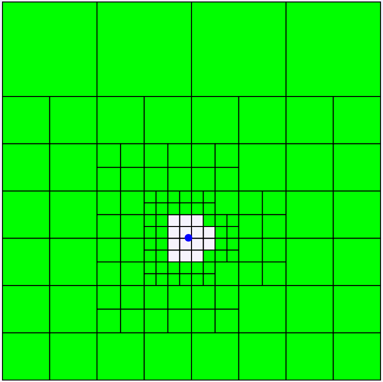
Example: Recursion



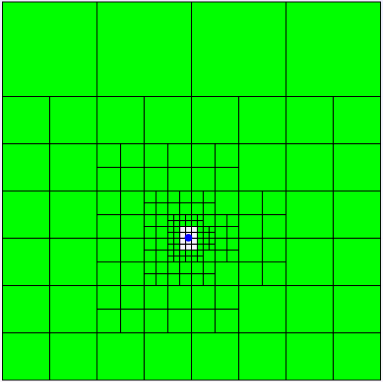
Example: Recursion



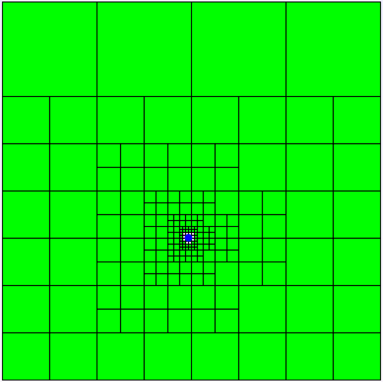
Example: Recursion



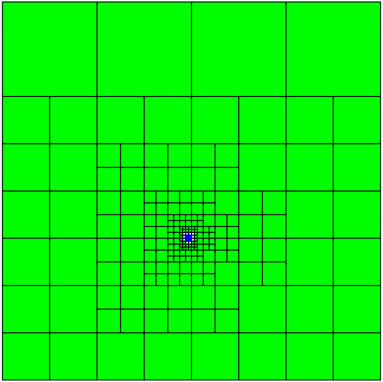
Example: Recursion



Example: Recursion



Example: Recursion



Observation: In typical situations, $\sim \log(n)$ clusters are sufficient.

Setup

Cluster tree: Recursive subdivision take $\sim n \log(n)$ operations.
Direct construction takes $\sim n$ operations, but is less flexible.

Coefficients: The masses of the "fictitious planets" can be computed directly in $\sim n\,k \log(n)$ operations.

$$z_{\sigma,\nu} = \sum_{j \in \hat{\sigma}} \ell_{\sigma,\nu}(y_j)\, m_j.$$

Setup

Cluster tree: Recursive subdivision take $\sim n \log(n)$ operations. Direct construction takes $\sim n$ operations, but is less flexible.

Coefficients: The masses of the "fictitious planets" can be computed directly in $\sim n\,k \log(n)$ operations.

$$z_{\sigma,\nu} = \sum_{j \in \hat{\sigma}} \ell_{\sigma,\nu}(y_j)\, m_j.$$

Forward transformation: If we use the same order for all clusters, we have

$$\ell_{\sigma,\nu}(y) = \sum_{\nu' \in M} \ell_{\sigma,\nu}(\xi_{\sigma',\nu'})\, \ell_{\sigma',\nu'}(y),$$

$$z_{\sigma,\nu} = \sum_{\sigma' \in \mathrm{chil}(\sigma)} \sum_{\nu' \in M} \ell_{\sigma,\nu}(\xi_{\sigma',\nu'}) \sum_{j \in \hat{\sigma}'} \ell_{\sigma',\nu'}(y_j)\, m_j$$

Setup

Cluster tree: Recursive subdivision take $\sim n\log(n)$ operations.
Direct construction takes $\sim n$ operations, but is less flexible.

Coefficients: The masses of the "fictitious planets" can be computed directly in $\sim n\,k\,\log(n)$ operations.

$$z_{\sigma,\nu} = \sum_{j\in\hat\sigma} \ell_{\sigma,\nu}(y_j)\, m_j.$$

Forward transformation: If we use the same order for all clusters, we have

$$\begin{aligned}
\ell_{\sigma,\nu}(y) &= \sum_{\nu'\in M} \ell_{\sigma,\nu}(\xi_{\sigma',\nu'})\, \ell_{\sigma',\nu'}(y),\\
z_{\sigma,\nu} &= \sum_{\sigma'\in\mathrm{chil}(\sigma)} \sum_{\nu'\in M} \ell_{\sigma,\nu}(\xi_{\sigma',\nu'}) \sum_{j\in\hat\sigma'} \ell_{\sigma',\nu'}(y_j)\, m_j\\
&= \sum_{\sigma'\in\mathrm{chil}(\sigma)} \sum_{\nu'\in M} \ell_{\sigma,\nu}(\xi_{\sigma',\nu'})\, z_{\sigma',\nu'}.
\end{aligned}$$

Setup

Cluster tree: Recursive subdivision take $\sim n \log(n)$ operations. Direct construction takes $\sim n$ operations, but is less flexible.

Coefficients: The masses of the "fictitious planets" can be computed directly in $\sim n\,k \log(n)$ operations.

$$z_{\sigma,\nu} = \sum_{j \in \hat\sigma} \ell_{\sigma,\nu}(y_j)\, m_j.$$

Forward transformation: If we use the same order for all clusters, we have

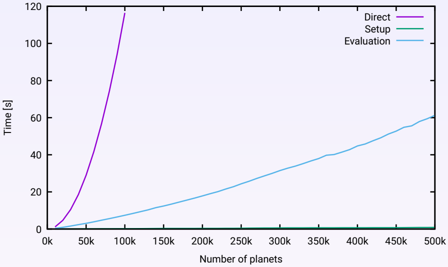
$$\ell_{\sigma,\nu}(y) = \sum_{\nu' \in M} \ell_{\sigma,\nu}(\xi_{\sigma',\nu'})\, \ell_{\sigma',\nu'}(y),$$

$$\begin{aligned} z_{\sigma,\nu} &= \sum_{\sigma' \in \mathrm{chil}(\sigma)} \sum_{\nu' \in M} \ell_{\sigma,\nu}(\xi_{\sigma',\nu'}) \sum_{j \in \hat\sigma'} \ell_{\sigma',\nu'}(y_j)\, m_j \\ &= \sum_{\sigma' \in \mathrm{chil}(\sigma)} \sum_{\nu' \in M} \ell_{\sigma,\nu}(\xi_{\sigma',\nu'})\, z_{\sigma',\nu'}. \end{aligned}$$

"Recycling" the childrens' masses reduces work to $\sim n\,k$ operations.

Experiment: Tree evaluation

Approach: Setup via forward transformation, recursive evaluation.



Observation: Recursive evaluation far more efficient.

Symmetric approximation

Goal: Reduce the runtime even further.

Assumption: All evaluation points $x_1, \ldots, x_n$ are known in advance.
$\to$ Construct clusters τ for these points, too.

Idea: Interpolate in both variables.

$$g(x,y) \approx \sum_{\nu \in M} \sum_{\mu \in M} g(\xi_{\tau,\nu}, \xi_{\sigma,\mu})\, \ell_{\tau,\nu}(x)\, \ell_{\sigma,\mu}(y)$$

Admissibility: We can expect a good accuracy if

$$\operatorname{diam}(\tau) \leq \eta \operatorname{dist}(\tau,\sigma) \qquad \text{and} \qquad \operatorname{diam}(\sigma) \leq \eta \operatorname{dist}(\tau,\sigma).$$

Three-phase algorithm

Forward transformation: For all source clusters σ, compute

$$z_{\sigma,\mu} = \sum_{j \in \hat{\sigma}} \ell_{\sigma,\mu}(y_j)\, m_j.$$

Recursion: If τ and σ are admissible, update

$$z_{\tau,\nu} \leftarrow z_{\tau,\nu} + \sum_{\mu \in M} g(\xi_{\tau,\nu}, \xi_{\sigma,\mu})\, z_{\sigma,\mu}.$$

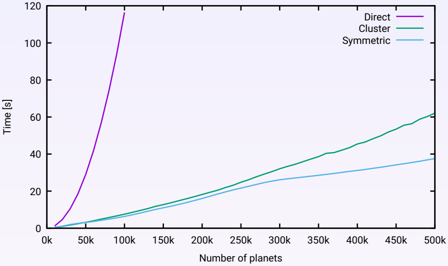
If τ and σ are inadmissible, switch to their children.
If there are no children, evaluate directly.

Backward transformation: For all target clusters τ, update

$$\varphi(x_i) \leftarrow \varphi(x_i) + \sum_{\nu \in M} \ell_{\tau,\nu}(x_i)\, z_{\tau,\nu}.$$

Experiment: Symmetric evaluation

Approach: Forward transformation, recursive symmetric interpolation, backward transformation.



Observation: Symmetric approach grows more efficient as the number of planets increases, requires $\sim n\,k$ operations.

Summary

Low-rank approximation allows us to prepare quantities in advance and re-use them for multiple evaluations of the potential.

Interpolation offers a simple way to obtain low-rank approximations. Can be interpreted as using "fictitious planets" to replace entire clusters.

Recursive evaluation based on an admissibility condition takes only $\sim \log(n)$ operations.

Symmetric approach improves the performance if all evaluation points are known in advance and can be clustered, as well.