FastMMLib: a generic Fast Multipole Method library

Eric DARRIGRAND

joint work with Yvon LAFRANCHE

IRMAR – Université de Rennes 1

First of all: definition of FMM boxes ?

Secondly: This is not about math developments, but C++ developments.
However, don't be scared ! Stay here ! Outside, there is ...

Outline

- Mathematical form of the FMM in FastMMLib
- Specificities of FastMMLib
- Structure of the library
- The user's classes
- State of the art

Mathematically

Efficient evaluation of matrix-vector products when the matrix is defined by a kernel or a potential G with a matrix $[M]$ of size $N \times N$: e.g.

$$[M]_{i\,j} = \int_\Gamma \int_\Gamma G(x,y)\varphi_j(y)\varphi_i(x)d\gamma(y)d\gamma(x)\,,$$

or

$$[M]_{i\,j} = G(x_i, y_j)\,,$$

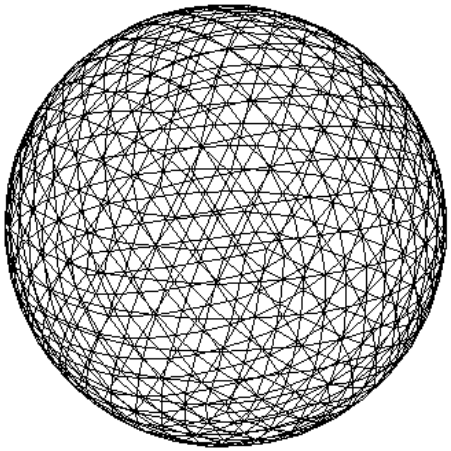
with complexity $\mathcal{O}(N^{3/2})$, $\mathcal{O}(N^{4/3})$ or $\mathcal{O}(N \log N)$.

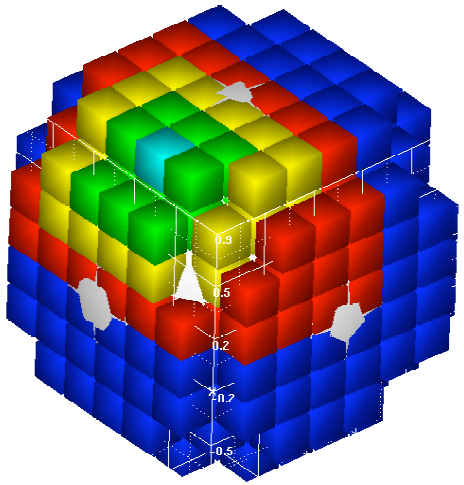
The essential ingredient is the choice of an expansion of the interaction function G:

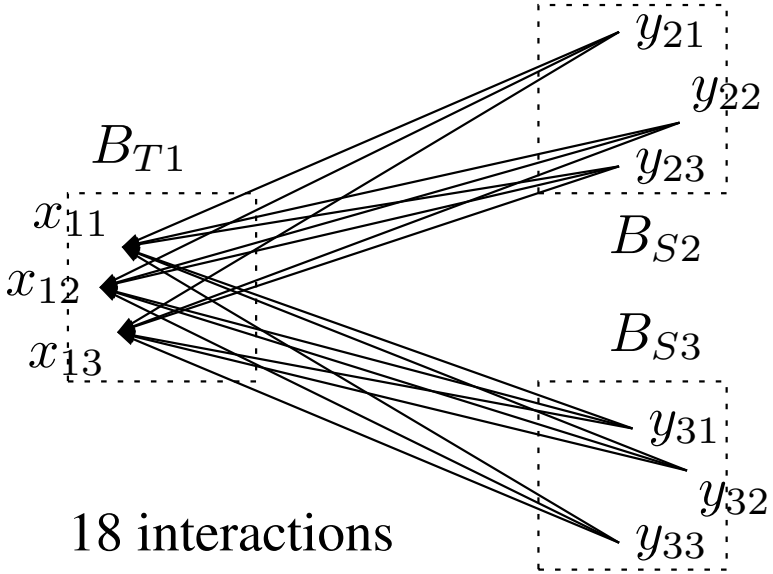
$$G(x,y) \approx \sum_{p=1}^{S} c_p \, g_{x,B}^{(p)} \sum_{\tilde{p}=1}^{\widetilde{S}} \mathcal{T}_{B,\widetilde{B}}^{(p,\tilde{p})} \, f_{y,\widetilde{B}}^{(p,\tilde{p})} \, ,$$

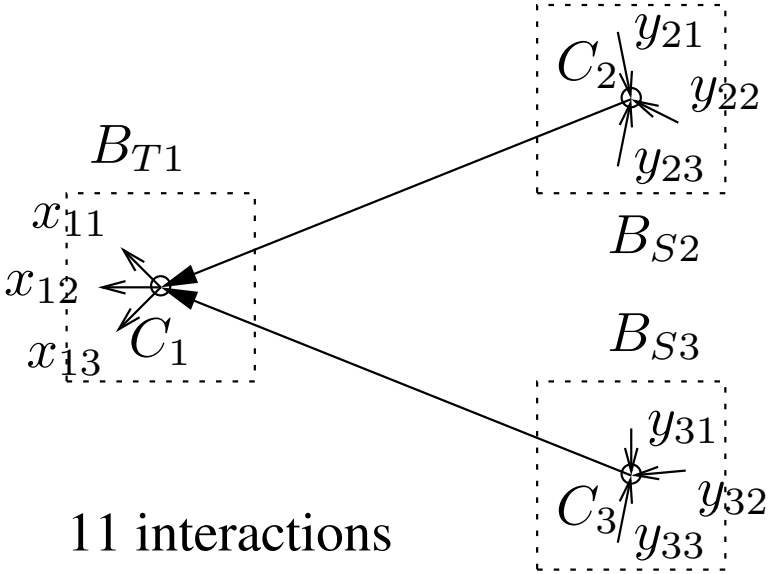
where

- S and $\widetilde{S}$: truncation or discretization parameters;
- c_p depends only on p;
- $f_{y,\widetilde{B}}^{(p,\tilde{p})}$ named *far moment*;
- $\mathcal{T}_{B,\widetilde{B}}^{(p,\tilde{p})}$ named *translation operator*, with B and $\widetilde{B}$ containing resp. x and y;
- $g_{x,B}^{(p)}$ named *local moment*.






y_{21}
y_{22}
y_{23}
B_{T1}
x_{11}
x_{12}
x_{13}
B_{S2}
B_{S3}
y_{31}
y_{32}
y_{33}
18 interactions


y_{21}
C_2
y_{22}
y_{23}
B_{T1}
x_{11}
x_{12}
C_1
x_{13}
B_{S2}
B_{S3}
y_{31}
C_3
y_{32}
y_{33}
11 interactions

Case of Helmholtz Green kernel in the framework of integral operators: e.g.

$$[M]_{i\,j} = \int_\Gamma \int_\Gamma G(x,y)\varphi_j(y)\varphi_i(x)d\gamma(y)d\gamma(x), \qquad \forall i,j$$

where

- $\star$ G is the Helmholtz fundamental solution, $G(x,y) = \frac{e^{ik\|x-y\|}}{4\pi\|x-y\|}$,
- $\star$ k the wavenumber,
- $\star$ $(\varphi_i)_i$ a set of finite-element basis functions.

An efficient calculation of the matrix-vector product $[M]\mathbf{X}$ given thanks to such an expansion "for i far from j":

$$[M]_{i\,j} \approx \sum_{p=1}^{S} c_p \sum_{B/B\cap \mathrm{supp}\varphi_i \neq \emptyset} g_{i,B}^{(p)} \sum_{\widetilde{B}/\widetilde{B}\cap \mathrm{supp}\varphi_j \neq \emptyset} \mathcal{T}_{B,\widetilde{B}}^{(p)}\, f_{j,\widetilde{B}}^{(p)},$$

with

$$c_p = \frac{ik}{(4\pi)^2} w_p \,,$$

$$g_{i,B}^{(p)} = \int_{B\cap \mathrm{supp}\varphi_i} e^{ik<s_p, x-C_B>} \varphi_i(x) d\gamma(x) \,,$$

$$f_{j,\widetilde{B}}^{(p)} = \int_{\widetilde{B}\cap \mathrm{supp}\varphi_j} e^{-ik<s_p, y-C_{\widetilde{B}}>} \varphi_j(y) d\gamma(y) \,,$$

and $\mathcal{T}_{B,\widetilde{B}}^{(p)}$ is the translation operator from the FMM box $\widetilde{B}$ to the FMM box B ...

... the translation operator is given by the expression

$$\mathcal{T}^{(p)}_{B,\widetilde{B}} = \sum_{\ell=1}^{L} (-i)^{\ell}(2\ell+1) h^{(1)}_{\ell}(k|C_B - C_{\widetilde{B}}|) P_{\ell}(\cos(s_p, C_B - C_{\widetilde{B}})),$$

and

- w_p, s_p are the quadrature rule on the unit sphere due to Funk-Hecke formula,
- "$\sum_{p=1}^{S}$" comes from the discretization of the Funk-Hecke formula,
- "$\sum_{\ell=1}^{L}$" is a truncation of the Gegenbauer series,
- C_B is the center of the FMM box B,
- $h^{(1)}_{\ell}$ is the spherical Hankel function of the first kind of degree ℓ,
- P_{ℓ} is the Legendre polynomial of degree ℓ,
- L and S are estimated thanks to the empirical formulae

$$L = kd + C(kd)^{1/3} \qquad S = (L+1)(2L+1) \qquad \text{with } d = \text{ diam. of the boxes.}$$

Case of Coulomb potential in the framework of particles:

$$[M]_{i\,j} = G(x_i, y_j) \quad \text{with} \quad G(x,y) = \frac{1}{\|x-y\|}.$$

"For i far from j": $[M]_{i\,j} \approx \sum_{l=0}^{L} \sum_{k=-l}^{l} g_{i,B}^{(l,k)} \sum_{n=0}^{\widetilde{L}} \sum_{m=-n}^{n} \mathcal{T}_{B,\widetilde{B}}^{((l,k),(n,m))} \; f_{j,\widetilde{B}}^{(n,m)}$,

where

$$g_{i,B}^{(l,k)} = \rho_i^l \; Y_l^k(\alpha_i, \beta_i) \quad \text{and} \quad f_{j,\widetilde{B}}^{(n,m)} = \rho_j^n \; Y_n^{-m}(\alpha_j, \beta_j)\,,$$

with $(\rho_i, \alpha_i, \beta_i) \leftrightarrow (x_i - C_B)$ and $(\rho_j, \alpha_j, \beta_j) \leftrightarrow (y_j - C_{\widetilde{B}})$,

and

$$\mathcal{T}_{B,\widetilde{B}}^{((l,k),(n,m))} = \frac{i^{|k-m|-|k|-|m|} \; A_n^m \; A_l^k}{(-1)^n \, A_{l+n}^{m-k}} \; \frac{Y_{l+n}^{m-k}(\alpha_{st}, \beta_{st})}{\rho_{st}^{l+n+1}}$$

with $A_n^m = \frac{(-1)^n}{\sqrt{(n-m)!(n+m)!}}$ and $(\rho_{st}, \alpha_{st}, \beta_{st}) \leftrightarrow (C_B - C_{\widetilde{B}})$.

Specificities of FastMMLib

Aim: a generic FMM library written in C++

- Initial idea: the library should work with any expansion of the form

$$G(i,j) = \sum_{p=1}^{S} c_p \sum_{B/i\in B} g_{i,B}^{(p)} \sum_{\widetilde{B}/j\in\widetilde{B}} \sum_{\widetilde{p}=1}^{\widetilde{S}} \mathcal{T}_{B,\widetilde{B}}^{(p,\widetilde{p})} \; f_{j,\widetilde{B}}^{(p,\widetilde{p})} \;\; ,$$

for all (i,j) such that i far from j;

... should deal with particles or finite-elements d.o.f.;

... should integrate the regularized form (RFMM);

> P. CHARTIER, E. D., E. FAOU. A regular fast multipole method for geometric numerical integrations of Hamiltonian systems. *BIT*, 50(1):23–40, 2010.

... should enable a particle or a d.o.f. to belong to several boxes.

- A FMM box may be identified by "several centroids".

> L. YING, G. BIROS, D. ZORIN. A kernel-independent adaptive fast multipole algorithm in two and three dimensions. *J. Comput. Phys.*, 196:591–626, 2004.

Crucial choices

• The library deals with the translations, the geometrical structure, the interpolations, ... the organization of the fast calculation.

  ◦ FastMMLib creates a geometrical object and provides usual Green kernels;

  ◦ The user can specify his own kernel (and then the expansion);

  ◦ FastMMLib provides a generic particle structure;

  ◦ The user should specify his own particles.

• To take in consideration the user's context (particles, FE-d.o.f.).

The far moments and local moments are calculated using user's functions:

$$g_{i,B}^{(l,k)} = \rho_i^l \, Y_l^k(\alpha_i, \beta_i) \qquad \text{with } (x_i - C_B) \leftrightarrow (\rho_i, \alpha_i, \beta_i)$$

or

$$g_{i,B}^{(p)} = \int_{B \cap \mathrm{supp}\varphi_i} e^{ik < s_p, x - C_B >} \varphi_i(x) d\gamma(x) \,,$$

the same for the close interactions.

Structure of FastMMLib

- ⋆ Classes related to the geometry:
  particles, FMM boxes, levels of the oc-tree
- ⋆ Classes related to the physics:
  Green kernels
- ⋆ Classes defining the FMM object:
  far/local moments, far/local fields, translation operators, FMM product

User-connected classes in FastMMLib

- ⋆ Geometry:
  - ◇ `NVParticle` and `IEMParticle` which derive from `Particle`,
  - ◇ `SetOfParticles`, `ListOfLevelsParameters`, `Levels`.
- ⋆ Physics: `Helm3DKernelGeg` which derives from `GreenKernel`.
- ⋆ FMM object: `FMM`
  - ◇ precalculation: call of `MomentContribution`, `TransferOperator`, `CloseInteraction`,
  - ◇ matrix-vector product: call of `FieldContribution`.

The classes `NVParticle` and `IEMParticle : public Particle`

- Data members

```
int tag_;
```

// numbering in the corresponding initial geometry.

```
const fmm_real_t* centroid_;
```

// centroid of the geometrical particle.

```
fmm_real_t diam_;
```

// charateristic length of the particle.

- Member functions

```
NVParticle(fmm_real_t* centroid, fmm_real_t diam);
```

// constructor;

```
const fmm_real_t* centroid(); fmm_real_t diam();
bool contained(const fmm_real_t* box_centroid,
               fmm_real_t box_diam );
```

.../...

```
void MomentComputation(
          void f(const fmm_real_t* y, fmm_complex_t* res),
          fmm_complex_t* res );
```

// define the operator:

$$f \mapsto f(\texttt{this->centroid()})$$

or

$$f \mapsto \left(\int_{B \cap \mathrm{supp}\varphi_i} f(x)\varphi_i(x)d\gamma(x) = \sum_{K \in B \cap \mathrm{supp}\varphi_i} \int_K f(x)\varphi_i(x)d\gamma(x) \right)_B$$

with f the function associated with the definition of the moments of the kernel expansion, e.g.

$$f(x) = \rho_x^l \; Y_l^k(\alpha_x, \beta_x) \qquad \text{with } (x - C_B) \leftrightarrow (\rho_x, \alpha_x, \beta_x)$$

$$\text{or} \qquad f(x) = e^{ik<s_p, x - C_B>}$$

The class `ListOfLevelsParameters`:

- Data members

```
fmm_real_t magnitude_exponent_;
fmm_number_t min_level_, max_level_;
fmm_real_t initial_dilatation_;
fmm_number_t neighbor_order_;
fmm_real_t rfmm_dilatation_;
```

- Member functions

A unique constructor with default values

```
set...(), get...();
```

// set and access functions

The class `Helm3DKernelGeg : public GreenKernel`

- Data members from `GreenKernel`

```
const Levels& geom_levels_;
fmm_number_t space_dim_, far_dim_, local_dim_, close_dim_;
KernelFMMFormat kernel_format_;
vector<fmm_number_t> S_, St_;
```

- Specific data members

```
fmm_real_t wave_number_;
fmm_number_t truncation_parameter_;
```

// coefficient to define the truncation parameter of the Gegenbauer series.

```
fmm_real_t* vx_y_;
```

// local working variable.

```
std::vector<std::vector<fmm_real_t> > sphere_directions_;
```

// for the Funk-Hecke formula.

The class `Helm3DKernelGeg : public GreenKernel`

- Member functions

```
Helm3DKernelGeg(const Levels& userLevels,
                const fmm_real_t waveNumber,
                const fmm_real_t truncationCoeffLog=0,
                const fmm_real_t truncationCoeff1_3=2.6);
```

// a constructor.

.../...

```
void KernelDefFunc(const fmm_real_t* vx,
                   const fmm_real_t* vy, fmm_complex_t* res);
```

// defines the kernel function.

```
void FarMomentDefFunc(const fmm_number_t& nsp,
                   const fmm_real_t* rx, fmm_complex_t* res);
```

// defines the far-moment function.

```
void LocalMomentDefFunc(const fmm_number_t& nsp,
                   const fmm_real_t* rx, fmm_complex_t* res);
```

// defines the local-moment function.

```
void F2LtransOpDefFunc(const fmm_number_t& nsp,
                   const fmm_number_t& nDir, fmm_complex_t* res);
```

// defines the operator involved in the F2L transfers.

An example of use:

```
using namespace std;

#include "FMM.h++"

int main() {
...

vector<NVParticle*> MyParticles;

SetOfParticles MySetsofParticles(MyParticles, ...);

ListOfLevelsParameters MyListOfParams(...);

Levels MyLevels(MySetsofParticles, MyListOfParams);

GreenKernel MyKernel(MyLevels);

FMM MyFMM(MyKernel);
...

V = MyFMM.MatVec(U);
...

}
```

State of the art

- ⋆ Geometry: validated
- ⋆ Physics: `Helm3DKernelGeg` implemented, under validation
- ⋆ SLFMM object: `FMM` implemented, under validation.
- ⋆ MLFMM object: Some field constructors still have to be implemented.
- ⋆ A first public version expected for june 2016.

State of the art

⋆ What about transient problems ?

The FMM expansion derived in

> A. A. ERGIN, B. SHANKER, E. MICHIELSSEN. Fast Evaluation of Three-Dimensional Transient Wave Fields Using Diagonal Translation Operators.
> *J. Comput. Phys.*, 146:157–180, 1998.

is here

> Alternatively, the fields at the M_o observers can be evaluated using Eq. (45), which, for the source density expressed by (51), takes the form

$$\tilde{u}_l\left(\mathbf{r}^{o(i)}, t\right) = \sum_{n=0}^{M'} \sum_{m=-M_n}^{M_n} \delta\left(t - \hat{\mathbf{k}}_{nm} \cdot \tilde{\mathbf{r}}^{o(i)}/c\right) * \mathcal{T}_{nm}(t) * \sum_{j=1}^{M_s} \delta\left(t + \hat{\mathbf{k}}_{nm} \cdot \tilde{\mathbf{r}}^{s(j)}/c\right) * f_l^j(t), \tag{53}$$

for the evaluation this field

> The field at the ith observer is given by

$$u\left(\mathbf{r}^{o(i)}, t\right) = \sum_{j=1}^{M_s} \frac{f^j\left(t - \left|\mathbf{r}^{o(i)} - \mathbf{r}^{s(j)}\right|/c\right)}{4\pi\left|\mathbf{r}^{o(i)} - \mathbf{r}^{s(j)}\right|}.$$

State of the art

⋆ What else ?

Integration of High Order method

O. P. BRUNO, L. A. KUNYANSKY. A Fast High Order Algorithm for the Solution of Surface Scattering Problems: Basic Implementation, tests, and Applications. *J. Comput. Phys.*, 169(1):80–110, May 2001.

and H-matrices

W. HACKBUSCH. A Sparse Matrix Arithmetic Based on H-Matrices. Part I: Introduction to H-Matrices. *Computing*, 62:89–108, 1999.

Thank you for your attention

and many thanks again for this exceptional workshop