Fast methods for integral equations

Matthieu Aussal
Research engineer

Centre de Mathématiques Appliquées de l’École Polytechnique
Route de Saclay - 91128 Palaiseau CEDEX France

Journée de rentrée du CMAP - 03 octobre 2018
Introduction

Integral representation

Integral equations

Acceleration

GYPSILAB

Conclusion
Main research topics and collaborations

**Spatial audio (3D audio):**
- VST plug’in for sound engineers (F. Salmon, CNSMDP),
- Audio guidance for blind people (S. Ferrand, MCV),
- Classroom sonorization (*Mon cartable connecté*).

**Auralization:**
- Numerical room acoustics for archeology (R. Gueguen),
- Real-time room acoustic for virtual reality,
- Augmented reality for museum visit (PSC, MusiX)

**Finite and boundary element method (FEM, BEM):**
- Gypsilab : Matlab framework for fast prototyping (F. Alouges)
- Fast methods for High Performance Computing :
  DDM, H-Matrix, FFM, ray-tracing, etc.
- Applications (scattering, 3D audio, eeg, Stokes flows, etc.)
Past, present and future collaborations
Some selected softwares

MyBino (freeware) :
Professional VST plug’in for binaural rendering.

Just4rir (open-source) :
Blender plug’in for room acoustics by ray-tracing.

Gypsilab (open-source part) :
Matlab framework for fast FEM-BEM prototyping.

Gypsilab (private part) :
High-Performance Computing for FEM-BEM problems.
Introduction

Integral representation

Integral equations

Acceleration

GYPSILAB

Conclusion
A boundary problem for acoustic

The full space $\Omega$ is separated in two subspaces $\Omega^i$ and $\Omega^e$ by a boundary $\Gamma$, smooth and oriented along normal $n$.

For $u : \Omega \setminus \Gamma \to \mathbb{C}$, we define:

- $u^i = u|_{\Omega^i}$
- $u^e = u|_{\Omega^e}$,
- $\mu = [u] = u^i - u^e$,
- $\lambda = [\partial_n u] = \partial_n u^i - \partial_n u^e$.

**Note:** $\Gamma$ is a singularity of $\Omega \Rightarrow$ Distribution theory!
Theorem of integral representation

If \( u \) satisfies Helmholtz equation and Sommerfeld condition:

\[
\begin{aligned}
-(\Delta u^i + k^2 u^i) &= 0 \text{ in } \Omega^i, \\
-(\Delta u^e + k^2 u^e) &= 0 \text{ in } \Omega^e, \\
\lim_{r \to +\infty} r \left( \partial_r u^e +iku^e \right) &= 0,
\end{aligned}
\]

then \( u \) satisfies an integral representation \( \forall \mathbf{x} \in \Omega^i \cup \Omega^e \):

\[
\begin{aligned}
u(\mathbf{x}) &= \int_{\Gamma} G(\mathbf{x}, \mathbf{y}) \lambda(\mathbf{y}) d\Gamma_y - \int_{\Gamma} \frac{\partial G(\mathbf{x}, \mathbf{y})}{\partial n_\mathbf{y}} \mu(\mathbf{y}) d\Gamma_y,
\end{aligned}
\]

with \( G(\mathbf{x}, \mathbf{y}) \) stand for the oscillatory Green kernel:

\[
G(\mathbf{x}, \mathbf{y}) = \frac{e^{ik|\mathbf{x} - \mathbf{y}|}}{4\pi|\mathbf{x} - \mathbf{y}|}.
\]

**Note**: Volumic data are fully determined by surfacic unknowns.
Evaluation of boundary integrals

- Regular integration are done using a quadrature rule $(y_q, \gamma_q)_{1 \leq q \leq N_q}$ for the boundary $\Gamma$:

$$\int_{\Gamma} G(x, y) \lambda(y) d\Gamma_y \approx \sum_{q=1}^{N_q} \gamma_q G(x, y_q) \lambda_\phi(y_q).$$

- If necessary, singular integrations are done analytically... and it’s hard! (A. Lefebvre-Lepot)

**Note**: Regular integration can be seen as a discrete convolution on a non-uniform grid... **meshless technics**!
Numerical application

Integral representation of the elementary solution $E = -\frac{e^{ik|x|}}{4\pi|x|}$.

Boundary $\Gamma$ is a unit sphere, meshed with 2 000 triangles and wave number is fixed at $k = 5$:

$$u(x) = \int_{\Gamma} G(x, y) \partial_n E(y) d\Gamma_y - \int_{\Gamma} \frac{\partial G(x, y)}{\partial n_y} E(y) d\Gamma_y$$

Reference $E(x)$, computed analytically in full space.
Introduction

Integral representation

Integral equations

Acceleration

GYPSILAB

Conclusion
Calderon projectors

Following the theorem of integral representation, if \( u \) satisfy:

\[
u(x) = \int_{\Gamma} G(x, y) \lambda(y) d\Gamma_y - \int_{\Gamma} \frac{\partial G(x, y)}{\partial n_y} \mu(y) d\Gamma_y,
\]

then \( u \) is solution of Helmholtz equation. Boundary traces are given respectively by **interior** and **exterior** Calderon projectors \( \forall x \in \Gamma \):

\[
\begin{pmatrix}
u_i \\
\partial_i n u_i
\end{pmatrix} = \begin{pmatrix}
\frac{+ld}{2} - D \\
-N
\end{pmatrix}
\begin{pmatrix}
\frac{+ld}{2} + D^t
\end{pmatrix}
\begin{pmatrix}
\mu \\
\lambda
\end{pmatrix},
\]

\[
\begin{pmatrix}
u_e \\
\partial_i n u_e
\end{pmatrix} = \begin{pmatrix}
\frac{-ld}{2} - D \\
-N
\end{pmatrix}
\begin{pmatrix}
\frac{-ld}{2} + D^t
\end{pmatrix}
\begin{pmatrix}
\mu \\
\lambda
\end{pmatrix},
\]

\[
S \lambda(x) = \int_{\Gamma} G(x, y) \lambda(y) d_{\Gamma_y}, \quad D^t \lambda(x) = \int_{\Gamma} \partial_{n_x} G(x, y) \lambda(y) d_{\Gamma_y},
\]

\[
D \mu(x) = \int_{\Gamma} \partial_{n_y} G(x, y) \mu(y) d_{\Gamma_y}, \quad N \mu(x) = \int_{\Gamma} \partial_{n_x} \partial_{n_y} G(x, y) \mu(y) d_{\Gamma_y}.
\]
Integral formulation: Example with Dirichlet scattering

1. Formulate problem considering $u^e = u_{tot} - u_{inc}$:

\[
\begin{cases}
-(\Delta u^e + k^2 u^e) = 0 \quad \text{in } \Omega^e, \\
\lim_{r \to +\infty} r (\partial_r u^e + iku^e) = 0, \\
u_{tot} = 0 \iff u^e = -u_{inc} \quad \text{in } \Gamma.
\end{cases}
\]

2. Choose the right projector (according to normal $n$):

\[
\begin{pmatrix}
u^e \\
\partial_n u^e
\end{pmatrix} = \begin{pmatrix}
-\frac{ld}{2} - D & S \\
-N & -\frac{ld}{2} + D^t
\end{pmatrix} \begin{pmatrix}
\mu \\
\lambda
\end{pmatrix}.
\]

3. Choose a jump extension:

\[u_i = u^e \iff \mu = 0.\]

4. Solve integral equation on $\Gamma$:

\[S\lambda = u^e = -u_{inc}.\]

5. Compute the total field on $\Omega^e$:

\[u_{tot} = S\lambda + u_{inc}.\]
Evaluation of boundary operators

Considering a finite elements basis \((\phi_n(y))_{1 \leq n \leq N_{dof}}\) to represent boundary functions:

\[
\lambda(y) \sim \lambda_\phi(y) = \sum_{n=1}^{N_{dof}} \lambda_n \phi_n(y) \quad \forall y \in \Gamma,
\]

Integral operators can be discretized using one of the following methods:

Collocation : 
\[
S_{ij} = \int_\Gamma G(x_i, y) \phi_j(y) \, dy,
\]

Galerkin : 
\[
S_{ij} = \int_\Gamma \int_\Gamma \phi_i(x) G(x, y) \phi_j(y) \, dy.
\]

Note : Using a discrete quadrature \((y_q, \gamma_q)_{1 \leq q \leq N_q}\) of the boundary \(\Gamma\), both methods need the computation of a full matrix \(G_{xy}\).
Numerical application

Integral representation of the solution of an integral equation, satisfying \textbf{Dirichlet} condition. Boundary $\Gamma$ is a unit sphere, meshed with 2 000 triangles and wave number is fixed at $k = 5$:

\[ \mu = 0 \Rightarrow S\lambda = -u_{inc} \]
\[ \lambda = 0 \Rightarrow \left( -\frac{Id}{2} - D \right) \mu = -u_{inc} \]

\textbf{Note} : Galerkin method were used with $P_1$ elements.
Introduction

Integral representation

Integral equations

Acceleration

GYPSILAB

Conclusion
Example with a regular kernel

All integral formulations need discrete convolutions with Green kernel on non-uniform grids \((x_i)_{i \in [1, N_x]}\) and \((y_j)_{j \in [1, N_y]}\) in \(\mathbb{R}^3\):

\[
u_i = \sum_{j=1}^{N_y} G(x_i, y_j)v_j.
\]

For example, considering a regular Green kernel:

\[
K(x, y) = |x - y|^2 = |x|^2 - 2x \cdot y + |y|^2,
\]

Convolution by \(K\) can be rewritten for all \(i\):

\[
u_i = |x_i|^2 \sum_{j=1}^{N_y} v_j - 2x_i \cdot \sum_{j=1}^{N_y} y_j v_j + \sum_{j=1}^{N_y} |y_j|^2 v_j
\]

Separate \(x\) and \(y\): Complexity \(O(N_x N_y) \rightarrow O(N_y) + O(N_x)!\)
Example with a singular kernel

**Context:** \( G(x, y) = \frac{e^{ik|x-y|}}{4\pi|x-y|} \) is **locally** singular (\(|x - y| < \epsilon\)).

**Idea:** Use "smoothing" properties issued from far distances.

**Method:** *Divide and conquer* with distance criterion.

**Algorithm(s):** Hierarchical tree and leaves compression.

**Initialization:** Considering a space sampling \((x_i)_{i \in [1,N_x]}\) and \((y_j)_{j \in [1,N_y]}\) in \(\mathbb{R}^3\), a Green kernel matrix is given by:

\[
G_{xy} = \begin{bmatrix}
G(x_1, y_1) & \cdots & G(x_1, y_{N_y}) \\
\vdots & \ddots & \vdots \\
G(x_{N_x}, y_1) & \cdots & G(x_{N_x}, y_{N_y})
\end{bmatrix} = [G_0]
\]
Hierarchical tree

Step 0

$G_{xy} = [G_0]$
Hierarchical tree

Step 1

\[ G_{xy} = \begin{bmatrix} \frac{G_1}{G_3} & \frac{G_2}{G_4} \end{bmatrix} \]
Hierarchical tree

Step 2

At level $n$, each $n$-tuple is unique and define the block hierarchy.

$$G_{xy} = \begin{bmatrix}
G_{11} & G_{12} & G_{21} & G_{22} \\
G_{13} & G_{14} & G_{21} & G_{22} \\
G_{31} & G_{32} & G_{41} & G_{42} \\
G_{33} & G_{34} & G_{43} & G_{44}
\end{bmatrix}$$
Hierarchical tree

Step 3

\[ G_{xy} = \begin{bmatrix}
G_{111} & G_{112} & \cdots & G_{21} & G_{22} \\
G_{121} & G_{122} & \cdots & G_{22} & G_{222} \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
G_{331} & G_{332} & \cdots & G_{441} & G_{442} \\
G_{333} & G_{334} & \cdots & G_{444} & G_{444}
\end{bmatrix} \]

Green leaves are far away \(\Rightarrow\) no recursion.
Hierarchical tree

Step 4
Hierarchical tree

Step 5
Hierarchical tree

Step 6
Hierarchical tree

Step 7
Hierarchical tree

and finally ...
Hierarchical tree

The last level is reached!

Green cells $\rightarrow$ far interactions
Red cells $\rightarrow$ close interactions
Leaves compression

Idea : Use "smoothing" properties issued from far distances to interpolate interactions $\rightarrow$ data compression!

Method : Considering $G_{ij} = (G(x_i, y_j))_{i \in I, j \in J}$ a block of far interactions, indexed by $I \in [1, N_x]$ and $J \in [1, N_y]$ :

- **Weak** interpolation :
  
  $$G(x, y) = \sum_{k} \mathcal{I}_k(x, \xi_k) G(\xi_k, y) \quad \Rightarrow \quad G_{IJ} = A_{Ik} B_{kJ}.$$

- **Strong** interpolation :
  
  $$G(x_i, y_j) = \sum_{k,l} \mathcal{I}_k(x, \xi_k) G(\xi_k, \xi_l) \mathcal{I}_l(\xi_l, y) \quad \Rightarrow \quad G_{IJ} = A_{Ik} T_{kl} B_{IJ}.$$
Final matrix form

Full

Weak

Strong
Non exhaustive list of fast solvers

<table>
<thead>
<tr>
<th>Method</th>
<th>Class</th>
<th>Interpolation</th>
<th>Complexity</th>
<th>Algebra</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full</td>
<td>-</td>
<td>-</td>
<td>$N^2$</td>
<td>(yes)</td>
</tr>
<tr>
<td>$\mathcal{H}^1$-matrix</td>
<td>weak</td>
<td>SVD/ACA</td>
<td>$N^{3/2}$</td>
<td>yes</td>
</tr>
<tr>
<td>$\mathcal{H}^2$-matrix</td>
<td>strong</td>
<td>Lagrange</td>
<td>$N \log N$</td>
<td>yes/no</td>
</tr>
<tr>
<td>SCSD</td>
<td>strong</td>
<td>Cardinal sine</td>
<td>$N^{6/5}$</td>
<td>no</td>
</tr>
<tr>
<td>FMM/FFM</td>
<td>strong</td>
<td>Plane wave</td>
<td>$N \log N$</td>
<td>no</td>
</tr>
</tbody>
</table>

- **SVD/ACA**: $G = AB^t$ with $A$ and $B$ low-rank matrices,
- **Lagrange**: $G = A \ast T \ast B^t$, $A$ and $B$ low-rank and $T$ small,
- **Plane wave**: $G(x, y) = \frac{ik}{16\pi^2} \lim_{L \to +\infty} \int_{s \in S^2} e^{iks \cdot x} M_1 T^L M_1 M_2(s) e^{iks \cdot y} ds$
- **Cardinal sine**: $G(x, y) = \frac{k}{(4\pi)^2} \sum_{n=0}^{+\infty} \alpha_n \rho_n \int_{S^2} e^{ik\rho_n s \cdot x} e^{-ik\rho_n s \cdot y} ds$
Context: For all particles \((x_i)_{i \in [1,N]}\) and \((y_j)_{j \in [1,N]}\) in \(\mathbb{R}^3\):

\[
u(x_i) = \frac{1}{4\pi|x_i - y_j|} \nu(y_j),
\]

\(\Leftrightarrow U = M \ast V.\)

with \(M \in M^N(\mathbb{R})\), \(U\) and \(V \in \mathbb{R}^N\).

Test case: \((x_i)_{i \in [1,N]}\) and \((y_j)_{j \in [1,N]}\) are uniformly distributed on the unit sphere \(S^2\).

Computer: 12 core at 2.7 GHz, 256 GO de ram, Matlab R2014a.
Numerical Application

<table>
<thead>
<tr>
<th>$N$</th>
<th>Time (s)</th>
<th>Memory peak</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10^4$</td>
<td>2.04</td>
<td>1 Mo</td>
</tr>
<tr>
<td>$10^5$</td>
<td>328</td>
<td>10 Mo</td>
</tr>
</tbody>
</table>

**Full product** (single thread)

<table>
<thead>
<tr>
<th>$N$</th>
<th>Build time (s)</th>
<th>MV time (s)</th>
<th>LU time (s)</th>
<th>Mem</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10^4$</td>
<td>8.24</td>
<td>0.16</td>
<td>9.88</td>
<td>100 Mo</td>
</tr>
<tr>
<td>$10^5$</td>
<td>78.9</td>
<td>1.68</td>
<td>145</td>
<td>1 Go</td>
</tr>
<tr>
<td>$10^6$</td>
<td>1069</td>
<td>25</td>
<td>4120</td>
<td>10 Go</td>
</tr>
</tbody>
</table>

$H^1$-matrix product and algebra with $\epsilon = 10^{-3}$ (single-thread)
## Numerical Application

<table>
<thead>
<tr>
<th>$N$</th>
<th>Time 1 core (s)</th>
<th>Time 12 cores (s)</th>
<th>Error L2</th>
<th>Mem</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10^4$</td>
<td>2.04</td>
<td>9.08</td>
<td>$8.03 \times 10^{-5}$</td>
<td>1 Mo</td>
</tr>
<tr>
<td>$10^5$</td>
<td>9.30</td>
<td>17.1</td>
<td>$1.34 \times 10^{-4}$</td>
<td>10 Mo</td>
</tr>
<tr>
<td>$10^6$</td>
<td>87.8</td>
<td>33.4</td>
<td>$1.35 \times 10^{-4}$</td>
<td>100 Mo</td>
</tr>
<tr>
<td>$10^7$</td>
<td>1063</td>
<td>169</td>
<td>$1.98 \times 10^{-4}$</td>
<td>1 Go</td>
</tr>
<tr>
<td>$10^8$</td>
<td>-</td>
<td>1499</td>
<td>$1.81 \times 10^{-4}$</td>
<td>10 Go</td>
</tr>
<tr>
<td>$10^9$</td>
<td>-</td>
<td>11340</td>
<td>$3.11 \times 10^{-4}$</td>
<td>100 Go</td>
</tr>
</tbody>
</table>

**Fast and Furious product with** $\epsilon = 10^{-3}$ (multi-thread)

**Note:** Assembly, full matrix would weigh **8 exa-octets**!
(double precision : $8N^2 \rightarrow 8 \times 10^{18}$ Bytes)
Introduction

Integral representation

Integral equations

Acceleration

**GYPsilab**

Conclusion
Learn to:

- Add, subtract, multiply and divide with confidence
- Deal with decimals, tackle fractions and make sense of percentages
- Size up weights, measures and shapes
- Prepare effectively for maths tests

By Matthieu Aussal
But what is GYPsilab?

GYPsilab is an **open-source** framework for **Matlab**, freely available under GPL 3.0, composed of object-oriented libraries:

- **openMsh**: Multi-dimensional mesh management.

- **openDom**: Numerical integration and variational calculation, *joint work with François Alouges*.

- **openFem**: Finite Element method for 2D and 3D shapes, *joint work with François Alouges*.

- **openHmx**: $H$-Matrix compression and algebra.

- **openRay**: Accelerated ray-tracing for HF problems, *joint work with Robin Gueguen*.

Designed to combine industrial requirements with fast prototyping.
Licensed add-on for **Gypsilab**

Open source part of **Gypsilab** is accompanied by **licensed add-on**, also composed of object-oriented libraries:

- **libHbm**: Block matrix computation and algebra. *Domain Decomposition Method, ℋ-Matrix Parallelization, multi-physic coupling, out-of core computation, etc.*

Application 1: Head-Related Transfer Function (3D audio)

Helmholtz equation

Neumann boundary condition

$P^1$ finite element

$10^{-3}$ compression accuracy

50,000 dof at 8 kHz

$\mathcal{H}$-Matrix exact solver

*Figure* – Acoustic scattering by human head, mesh from SYMARE database.
Application 2 : Radar detection (ISAE 2016)

Maxwell equation
PEC boundary condition, CFIE
RWG finite element

$10^{-3}$ compression accuracy
$300,000$ dof at $2.5$ GHz
$\mathcal{H}$-Matrix exact solver

**Figure** – Electromagnetic scattering by an UAV, workshop ISAE, mesh from Onera.
Application 3: Sonar detection (BeTSSI)

Helmholtz equation
Neumann boundary condition
$P^1$ finite element

$10^{-3}$ compression accuracy
$6,000,000$ dof at $1600$ Hz
Prec. FFM, DDM (32 cores)
Application 3: Sonar detection (BeTSSI)

Helmholtz equation
Neumann boundary condition
$P^1$ finite element

$10^{-3}$ compression accuracy

6,000,000 dof at 1600 Hz
Prec. FFM, DDM (32 cores)
Application 4: Space launcher scattering

Maxwell equation
PEC boundary condition, CFIE
$RWG$ finite element

$10^{-3}$ compression accuracy
$60,000,000$ dof at $2$ GHz
Prec. FFM, DDM (32 cores)

Note: Without compression, full operator would weigh $57$ Po!
(complex double precision: $16N^2 \approx 5.7 \times 10^{16}$ Bytes)
Application 4: Space launcher scattering

Maxwell equation
PEC boundary condition, CFIE
$RWG$ finite element

$10^{-3}$ compression accuracy
$60.000.000$ dof at $2$ GHz
Prec. FFM, DDM (32 cores)

Note: Without compression, full operator would weigh $57$ Po!
(complex double precision: $16N^2 \approx 5.7 \times 10^{16}$ Bytes)
Application 4: Space launcher scattering

Maxwell equation
PEC boundary condition, CFIE
RWG finite element

$10^{-3}$ compression accuracy
$60.000.000$ dof at $2$ GHz
Prec. FFM, DDM (32 cores)

Note: Without compression, full operator would weigh 57 Po!
(complex double precision: $16N^2 \approx 5.7 \times 10^{16}$ Bytes)
Introduction

Integral representation

Integral equations

Acceleration

GYPSILAB

Conclusion


Conclusion

- Communications
- Kalman filtering with $\mathcal{H}$-Matrix (P. Moireau)
- Electro-encephalogram with $\mathcal{H}$-Matrix (A. Gramfort et al)
- Vibro-acoustic with FFM (Naval group)
- Add numericals methods (Finite volume, finite difference, etc).
- Mesh generation (2D, 3D, coupling)
- Reach world’s records ($3.10^9$ Maxwell, $70.10^9$ Stokes)

Download gypsilab: www.cmap.polytechnique.fr/~aussal

Il faut se débarrasser des casse-têtes.
On ne vit qu’une fois.

Charles Aznavour, 1924-2018