Fast Evaluation of Layer Potentials using Quadrature by Expansion for Photonics Applications

Andreas Klöckner

Computer Science · University of Illinois at Urbana-Champaign

December 15, 2016
Outline

1. Introduction
2. Developing QBX
3. Acceleration
4. Software
5. Results
Outline

1 Introduction
2 Developing QBX
3 Acceleration
4 Software
5 Results
Fundamental Solutions

Laplace Equation

\[ \nabla^2 u = 0 \]

Monopole
Fundamental Solutions

Laplace Equation

$$\Delta u = 0$$

Monopole

Helmholtz Equation

$$\Delta u + k^2 u = 0$$

Monopole
Fundamental Solutions

Laplace Equation

\[ \nabla u = 0 \]

Monopole

Dipole

Helmholtz Equation

\[ \nabla u + k^2 u = 0 \]

Monopole
Fundamental Solutions

Laplace Equation
\[ \Delta u = 0 \]
- Monopole
- Dipole

Helmholtz Equation
\[ \Delta u + k^2 u = 0 \]
- Monopole
- Dipole

Can take this arbitrarily far: Quadrupole, . . .
Fundamental Solutions

Laplace Equation

\[ \Delta u = 0 \]

Monopole

Dipole

Helmholtz Equation

Can take this arbitrarily far:

Quadrupole, …
Main question for numerical solution of PDEs:

How is the solution represented?

Our choice here:  
*Sums of fundamental solutions*

\[
\tilde{u}(x) = \sum_{i=1}^{N} G(|x - y_i|)\sigma_i
\]

- Is the solution reachable in this way?
  - Uniqueness?
- Linearity \(\rightarrow\) must satisfy PDE
- Boundary conditions: not necessarily
Summing Fundamental Solutions
Summing Fundamental Solutions
Summing Fundamental Solutions
Summing Fundamental Solutions
Summing Fundamental Solutions
Layer Potentials

\[(S\sigma)(x) := \int_{\Gamma} G_k(x - y)\sigma(y)ds_y \]

\[(D\sigma)(x) := PV \int_{\Gamma} n \cdot \nabla_y G_k(x - y)\sigma(y)ds_y \]

- Operators—map function \( \sigma \) on \( \Gamma \) to...
  - . . . function on \( \mathbb{R}^n \)
  - . . . function on \( \Gamma \) (in particular)
**Layer Potentials**

\[
(S\sigma)(x) := \int_{\Gamma} G_k(x - y)\sigma(y)\,ds_y \\
(D\sigma)(x) := PV \int_{\Gamma} n \cdot \nabla_y G_k(x - y)\sigma(y)\,ds_y
\]

- Operators—map function \(\sigma\) on \(\Gamma\) to...
  - ...function on \(\mathbb{R}^n\)
  - ...function on \(\Gamma\) (in particular)

To solve \(\triangle u = 0\) with \(u|_{\partial\Omega} = g\), find \(\sigma\) such that

\[
\lim_{x \to \partial\Omega-} (D\sigma)(x) = g
\]
Layer Potentials

\[(S\sigma)(x) := \int_{\Gamma} G_k(x - y)\sigma(y)ds_y\]

\[(D\sigma)(x) := PV \int_{\Gamma} n \cdot \nabla_y G_k(x - y)\sigma(y)ds_y\]

- Operators—map function \(\sigma\) on \(\Gamma\) to...
  - ...function on \(\mathbb{R}^n\)
  - ...function on \(\Gamma\) (in particular)

To solve \(\triangle u = 0\) with \(u|_{\partial\Omega} = g\), find \(\sigma\) such that

\[
\lim_{x \to \partial\Omega} \quad \text{Remaining difficulty: Quadrature}
\]
A Sampling of Ideas

- **Analytic/symbolic integration**
  - Sometimes possible
  - Problematic because of geometry description
  - Numerical stability of resulting formulas?

- **Adaptive integration**
  - Fails because (many) singularities are not integrable
  - Expensive

- **Change of variables/singularity subtraction/cancellation**
  - Algebraic trickery weakens/removes singularity
  - Not general-purpose (across dimensions, kernels)
A Sampling of Ideas

- **Analytic/symbolic integration**
  - Sometimes possible
  - Problematic because of geometry description
  - Numerical stability of resulting formulas?

- **Adaptive integration**
  - Fails because (many) singularities are not integrable
  - Expensive

- **Change of variables/singularity subtraction/cancellation**
  - Algebraic trickery weakens/removes singularity
  - Not general-purpose (across dimensions, kernels)

Primary drawbacks:
- Not generic in singularity
- Depend on curve (2D?) geometry
A Sampling of Ideas

- **Analytic/symbolic integration**
  - Sometimes possible
  - Problematic because of geometry description
  - Numerical stability of resulting formulas?

- **Adaptive integration**
  - Fails because (many) singularities are not integrable
  - Expensive

- Change of variables/singularity subtraction/cancellation
  - Algebraic trickery weakens/removes singularity
  - Not general-purpose (across dimensions, kernels)

Many more have worked on the problem: Sidi, Strain, Helsing, Davis, Duffy, Graglia, Hackbusch, Khayat, Schwab, Ying, Beale, Goodman, Haroldson, Lowengrub, Alpert, Rokhlin, Gimbutas, Bruno, Zorin, ...

Primary drawbacks:
- Not generic in singularity
- Depend on curve (2D?) geometry
Kernel Regularization

Singularity makes integration troublesome: *Get rid of it!*

\[
\sqrt{(x - y)^2} \rightarrow \sqrt{(x - y)^2 + \varepsilon^2}
\]

Use extrapolation to recover limit as \( \varepsilon \rightarrow 0 \).

(May also use geometric motivation: limit along line towards singular point.)

Primary drawbacks:

- Low-order accurate
- Need to make \( \varepsilon \) smaller (i.e. kernel more singular) to get better accuracy
Quadrature wish list

- Unstructured geometries
  - Adaptable to many engineering problems
  - Compatible with adaptive discretization
  - Leverage off-the-shelf mesh generation technology
- General-purpose
  (as much as possible—in kernel, singularity, . . . )
- Robust
- Well-conditioned
- High order
  - Also in geometry representation
- Compatible with FMM
Outline

1. Introduction

2. Developing QBX
   - Some intuition
   - Achieving high order
   - Other Potentials

3. Acceleration

4. Software

5. Results
Outline

1. Introduction

2. Developing QBX
   - Some intuition
   - Achieving high order
   - Other Potentials

3. Acceleration

4. Software

5. Results

Andreas Klöckner

Fast Evaluation of Layer Potentials using QBX
Using just the trapezoidal rule
Using just the trapezoidal rule

Effectively: replacing layer of charge with discrete sources.
Using just the trapezoidal rule:

- Pick a point off surface: \( c := x + \hat{n}r \) in ‘accurate’ region
- One-sided smooth potential: Field value there is an approximation to one-sided limit (Err=\(O(r)\))

Effectively: replacing layer of charge with discrete sources.
Using just the trapezoidal rule:

- Pick a point off surface: $c := x + \hat{n}r$
  - in ‘accurate’ region
- One-sided smooth potential: Field value there is an approximation to one-sided limit ($\text{Err} = O(r)$)

**But:** Can do much better!

Effectively: replacing layer of charge with discrete sources.
Fast Evaluation of Layer Potentials using QBX

Andreas Klöckner
Andreas Klöckner

Fast Evaluation of Layer Potentials using QBX
Fast Evaluation of Layer Potentials using QBX
Fast Evaluation of Layer Potentials using QBX

$p = 3, N = 80$

Some intuition  High order  DLP etc.
Some intuition

High order DLP etc.

\( p = 3, \ N = 80 \)

\( p = 6, \ N = 80 \)

\( p = 12, \ N = 80 \)
Andreas Klöckner

Fast Evaluation of Layer Potentials using QBX
Graf’s addition theorem
Graf’s addition theorem

\[ H_0^{(1)}(k|x - x'|) = \sum_{l=-\infty}^{\infty} H_l^{(1)}(k|x' - c|) e^{il\theta'} J_l(k|x - c|) e^{-il\theta} \]
Requires: $|x - c| < |x' - c|$ ("local expansion")

Graf’s addition theorem

\[ H_0^{(1)}(k|x - x'|) = \sum_{l=-\infty}^{\infty} H_l^{(1)}(k|x' - c|) e^{il\theta'} J_l(k|x - c|) e^{-il\theta} \]
Compute layer potential on the disk as

\[ S_k \sigma(x) = \sum_{l=-\infty}^{\infty} \alpha_l J_l(k\rho) e^{-il\theta} \]

with

\[ \alpha_l = \frac{i}{4} \int_{\Gamma} H_l^{(1)}(k|x'| - c|) e^{il\theta'} \sigma(x') \, dx' \quad (l = -\infty, \ldots, \infty) \]

\( S\sigma \) is a smooth function up to \( \Gamma \).
Compute layer potential on the disk as

\[ S_k \sigma(x) = \sum_{l=-\infty}^{\infty} \alpha_l J_l(k \rho) e^{-il\theta} \]

with

\[ \alpha_l = \frac{i}{4} \int_{\Gamma} H_l^{(1)}(k |x' - c|) e^{il\theta'} \sigma(x') \, dx' \quad (l = -\infty, \ldots, \infty) \]

\( S\sigma \) is a smooth function \textit{up to} \( \Gamma \).
Now discretize.

Compute layer potential on the disk as

$$S_k\sigma(x) = \sum_{l=-p}^{p} \alpha_l J_l(k\rho) e^{-il\theta}$$

with

$$\alpha_l = \frac{i}{4} \int_{\Gamma} H_l^{(1)}(k|x'| - c|) e^{il\theta'} \sigma(x') \, dx' \quad (l = -\infty, \ldots, \infty)$$

$S\sigma$ is a smooth function up to $\Gamma$. 
Compute layer potential on the disk as

\[ S_k \sigma(x) = \sum_{l=-p}^{p} \alpha_l J_l(k \rho) e^{-il\theta} \]

with

\[ \alpha_l = \frac{i}{4} T_N \left( \int_{\Gamma} H^{(1)}_l(k|x' - c|) e^{il\theta'} \sigma(x') \, dx' \right) \quad (l = -\infty, \ldots, \infty) \]

\( S \sigma \) is a smooth function up to \( \Gamma \).
Compute layer potential on the disk as

\[ S_k \sigma(x) = \sum_{l=-p}^{p} \alpha_l J_l(k \rho) e^{-il\theta} \]

with

\[ \alpha_l = \frac{i}{4} T_N \left( \int_{\Gamma} H_l^{(1)}(k |x' - c|) e^{il\theta'} \sigma(x') \, dx' \right) \quad (l = -\infty, \ldots, \infty) \]

\( S\sigma \) is a smooth function up to \( \Gamma \).

Two limits \( (p, N \to \infty) \)! Experiment showed: order matters!
Compute layer potential on the disk as

\[ S_k \sigma(x) = \sum_{l=-p}^{p} \alpha_l J_l(k \rho) e^{-i l \theta} \]

with

\[ \alpha_l = \frac{i}{4} T_N \left( \int_{\Gamma} H_l^{(1)}(k|x' - c|) e^{i l \theta'} \sigma(x') \, dx' \right) \quad (l = -\infty, \ldots, \infty) \]

Two limits \((p, N \to \infty)\)! Experiment showed: *order matters!*

*And*: failure and repair not actually surprising.
Outline

1. Introduction
2. Developing QBX
   - Some intuition
   - Achieving high order
   - Other Potentials
3. Acceleration
4. Software
5. Results
Achieving high order

Error $\leq \left( C \frac{r^{p+1}}{h} + C \frac{h^q}{r} \right) \|\sigma\|$ | Quadrature error
| Truncation error

Two approaches:

- **Asymptotically convergent**: $r = \sqrt{h}$
  - Error $\to 0$ as $h \to 0$
  - Low order: $h^{(p+1)/2}$

- **Convergent with controlled precision**: $r = 5h$
  - Error $\not\to 0$ as $h \to 0$
  - High order: $h^{p+1}$
  - to controlled precision $\varepsilon := (1/5)^q$
Outline

1. Introduction

2. Developing QBX
   - Some intuition
   - Achieving high order
   - Other Potentials

3. Acceleration

4. Software

5. Results
Other layer potentials

Can’t just do single-layer potentials:

\[ \alpha_l^P = \frac{i}{4} \int_{\Gamma} \frac{\partial}{\partial \hat{n}_{x'}} H_l^{(1)}(k|x' - c|) e^{i\theta'} \mu(x') \, dx'. \]

Even easier for target derivatives (\(S'\) et al.):

Take derivative of local expansion.

**Analysis says:** Will lose an order.
Other layer potentials

Can’t just do single-layer potentials:

$$\alpha^D_I = \frac{i}{4} \int_{\Gamma} \frac{\partial}{\partial \hat{n}_{x'}} H^{(1)}_I(k|x' - c|) e^{i\theta'} \mu(x') \, dx'.$$

Even easier for target derivatives (S’ et al.):

Take derivative of local expansion.

**Analysis says:** Will lose an order.

**Slight issue:** QBX computes one-sided limits.

Fortunately: Jump relations are known—e.g.

$$(PV)D^*\mu(x)|_{\Gamma} = \lim_{x^\pm \to x} D\mu(x^\pm) = \frac{1}{2} \mu(x).$$
Other layer potentials

Can’t just do single-layer potentials:

\[
\alpha^P = \frac{i}{4} \int_{\Gamma} \frac{\partial}{\partial \hat{n}_{x'}} H_l^{(1)}(k|x'| - c|) e^{il\theta'} \mu(x') \, dx'.
\]

Even easier for target derivatives (S’ et al.):

Take derivative of local expansion.

**Analysis says:** Will lose an order.

**Slight issue:** QBX computes one-sided limits.

Fortunately: Jump relations are known—e.g.

\[
(PV)D^* \mu(x)|_{\Gamma} = \frac{1}{2}
\]

**Alternative:** Two-sided average
Other layer potentials

Can’t just do single-layer potentials:

\[ \alpha^D_l = \frac{i}{4} \int_{\Gamma} \frac{\partial}{\partial \hat{n}_{x'}} H^{(1)}_l(k|x' - c|) e^{i\theta'} \mu(x') \, dx'. \]

Even easier for target derivatives (S’ et al.):

Take derivative of local expansion.

Analysis says: Will lose an order.

Slight issue: QBX computes one-sided limits.

Fortunately: Jump relations are known–e.g.

\[ (PV)D^* \mu(x)|_{\Gamma} = \lim_{x^+ \to x} D \mu(x^+) - \lim_{x^- \to x} D \mu(x^-) = \frac{1}{2}\mu(x). \]

Alternative: Two-sided average

Preferred because of conditioning
Outline

1. Introduction
2. Developing QBX
3. Acceleration
4. Software
5. Results
Fast Multipole Methods

\[ u(x) = \int_{\Gamma} G(x, x') \sigma(x') dS_{x'} \]

Only works if sources are 'far enough' away from targets.

Good: true for most particle pairs
Fast Multipole Methods

\[ u(x) = \int_{\Gamma} G(x, x') \sigma(x') dS_{x'} \]

Only works if sources are 'far enough' away from targets.

Good: true for most particle pairs
Fast Multipole Methods

\[ u(x) = \int_{\Gamma} G(x, x') \sigma(x') dS_{x'} \]

Only works if sources are ‘far enough’ away from targets.

Good: true for most particle pairs
Fast Multipole Methods

\[ u(x) = \int_{\Gamma} G(x, x') \sigma(x') dS_{x'} \]

Only works if sources are ‘far enough’ away from targets.

Good: true for most particle pairs
Fast Multipole Methods

$$u(x) = \int_{\Gamma} G(x, x') \sigma(x') dS_{x'}$$

Only works if sources are 'far enough' away from targets.

Good:
true for most particle pairs
Fast Multipole Methods

\[ u(x) = \int_{\Gamma} G(x, x') \sigma(x') dS_{x'} \]

Only works if sources are ‘far enough’ away from targets.

Good: true for most particle pairs
Fast Multipole Methods

\[ u(\mathbf{x}) = \int_{\Gamma} G(\mathbf{x}, \mathbf{x}') \sigma(\mathbf{x}') dS_{\mathbf{x}'} \]

Only works if sources are 'far enough' away from targets.

Good: true for most particle pairs
Fast Multipole Methods

\[ u(x) = \int_{\Gamma} G(x, x') \sigma(x') dS_{x'} \]

Only works if sources are 'far enough' away from targets.

Good: true for most particle pairs

Andreas Klöckner

Fast Evaluation of Layer Potentials using QBX
Fast Multipole Methods

\[ u(x) = \int_{\Gamma} G(x, x') \sigma(x') dS_{x'} \]

Only works if sources are 'far enough' away from targets.

Good: true for most particle pairs
Fast Multipole Methods

\[ u(x) = \int_{\Gamma} G(x, x') \sigma(x') dS_{x'} \]

Only works if sources are 'far enough' away from targets.

Good: true for most particle pairs

Andreas Klöckner
Fast Multipole Methods

\[ u(x) = \int_{\Gamma} G(x, x') \sigma(x') \, dS_{x'} \]

Only works if sources are 'far enough' away from targets.

Andreas Klöckner

Fast Evaluation of Layer Potentials using QBX
Fast Multipole Methods

\[ u(\mathbf{x}) = \int_{\Gamma} G(\mathbf{x}, \mathbf{x}') \sigma(\mathbf{x}') dS_{\mathbf{x}'} \]

Only works if sources are ‘far enough’ away from targets.

Good: true for most particle pairs
Fast Multipole Methods

$u(x) = \int G(x, x') \sigma(x') \, dS(x')$

Only works if sources are ‘far enough’ away from targets.

**Good:** true for most particle pairs
Fast Multipole Methods
Valid because all off-surface local expansions are actually expansions of the same smooth function—the layer potential.

Andreas Klöckner

Fast Evaluation of Layer Potentials using QBX
Valid because all off-surface local expansions are actually expansions of the same smooth function—the layer potential.
Valid because all off-surface local expansions are actually expansions of the same smooth function—the layer potential.

**True, but:**

- The FMM is not exact.  
  Multipole-to-Local translation incurs truncation error. This pollutes the higher-order coefficients.

- For evaluation of potential ($\sim$ mode 0) and gradient ($\sim$ mode 1) this is not significant.

- When asking for many high-order coefficients (as QBX does), there is an impact on accuracy.

**Remedy:** Use higher-order expansions in the FMM to push truncation order into higher-order coefficients.
Global QBX: Dealing with geometry
Global QBX: Dealing with geometry

Refinement moves center closer. → Problem mitigated.
Refinement moves center closer.
→ Problem mitigated.
Global QBX: Dealing with geometry

Refinement moves center closer. → Problem mitigated.

Remaining problem: Detect that an issue exists.
More subtle: Quadrature Requirements

- Mesh spacing $h_2$ on $\Gamma_2$ is not small enough to control the QBX quadrature error term. (Since it is larger than $2h_1$, which is the bound based on which we choose placement of centers.)
- Safe distance from $c$ is controlled by source element!
- Refinement (of $\Gamma_2$) is an effective remedy.
- **Again—Remaining problem:** Detect that an issue exists.
Area Queries

The boxes marked $p$ are the peers of the box marked $b$.

Enumeration of the peers of $b$ (chosen to have $\sim$ length scale as query box) and children yields leaf boxes covered by query square.

Area queries: Surprisingly versatile.

Example: To find ‘endangering’ sources, enumerate leaf boxes within endangered region around source. Tag leaf boxes with max. distance to endangering source. Now enumerate all leaf boxes overlapping QBX expansion disk to find max. radius of endangering source. Lastly: Find endangering sources by area query with found radius.
Area Queries

This improves on traversing from the root because (e.g.) a query square that overlaps a high-level boundary may have to traverse a subtree *disproportionately larger* than its own length scale.

Area queries: Surprisingly versatile. Example: To find 'endangering' sources, enumerate leaf boxes within endangered region around source. Tag leaf boxes with max. distance to endangering source. Now enumerate all leaf boxes overlapping QBX expansion disk to find max. radius of endangering source. Lastly: Find endangering sources by area query with found radius.
Area Queries

This improves on traversing from the root because (e.g.) a query box that overlaps a high-level boundary may have to traverse a portionately larger than its own length scale.

Area queries: Surprisingly versatile.

Example:

- To find ‘endangering’ sources, enumerate leaf boxes within endangered region around source.
- Tag leaf boxes with max. distance to endangering source.
- Now enumerate all leaf boxes overlapping QBX expansion disk to find max. radius of endangering source.
- Lastly: Find endangering sources by area query with found radius.
Barnett close eval + QBX in an FMM

Targets have ‘evaluation type’:

1. Volume
2. One-sided volume
3. One-sided self

Algorithm:

1. Build look-up table:
   - Box $\rightarrow$ overlapped QBX disks
2. For each target:
   1. Find containing box
      1. Find overlapping QBX disks
      2. If one found, require QBX use
      3. Find the one with the closest center (prefer global)
Barnett close eval + QBX in an FMM

Targets have ‘evaluation type’:

1. Volume
2. One-sided volume
3. One-sided self

Algorithm:

1. Build look-up table:
   - Box $\rightarrow$ overlapped QBX disks
2. For each target:
   1. Find containing box
   2. Find overlapping QBX disks
   3. If one found, require QBX
   4. Find the one with the closest center

→ unified handling of layer and volume targets
# QBX + FMM: Many possibilities

<table>
<thead>
<tr>
<th>“Global” QBX</th>
<th>“Local” QBX</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓ (In principle) Simple mod. to existing FMM</td>
<td>✓ Main ‘point’ FMM algorithm unchanged</td>
</tr>
<tr>
<td>✗ (In practice) Significant preprocessing required to ensure accuracy</td>
<td>■ Only modify near-field</td>
</tr>
<tr>
<td></td>
<td>✗ Sized targets make some FMM modification appropriate</td>
</tr>
<tr>
<td></td>
<td>✗ Requires preprocessing to determine quadrature/FMM near-field mismatch</td>
</tr>
<tr>
<td>✗ Requires FMM order increase</td>
<td>✓ Robust wrt geometry</td>
</tr>
<tr>
<td>✓ Comparatively cheap</td>
<td>✗ Comparatively expensive</td>
</tr>
<tr>
<td>✓ Fits directly into acceleration structure</td>
<td>✗ Does not require refinement</td>
</tr>
<tr>
<td>✓ Robust wrt geometry</td>
<td></td>
</tr>
</tbody>
</table>

- I.e. area queries and mesh refinement

- Requires FMM order increase
- Comparatively cheap
- Fits directly into acceleration structure
- Robust wrt geometry
- Comparatively expensive
- Does not require refinement

---

**Andreas Klöckner**  
Fast Evaluation of Layer Potentials using QBX
**QBX + FMM: Many possibilities**

### “Global” QBX

- ✓ (In principle) Simple mod. to existing FMM
- ❌ (In practice) Significant preprocessing required to ensure accuracy
  - I.e. area queries and mesh refinement
- ❌ Requires FMM order increase
- ✓ Comparatively cheap
- ✓ Fits directly into acceleration structure
- ✓ Robust wrt geometry

### “Local” QBX

- ✓ Main ‘point’ FMM algorithm unchanged
  - Only modify near-field
- ❌ Sized targets make some FMM modification appropriate
- ❌ Requires preprocessing to determine quadrature/FMM near-field mismatch
- ✓ Robust wrt geometry

Mainly: Cost tradeoff

Obvious idea: Hybrid scheme

Andreas Klöckner

Fast Evaluation of Layer Potentials using QBX
Outline

1. Introduction
2. Developing QBX
3. Acceleration
4. Software
5. Results
loopy

Code generator for CPUs/GPUs

Mathematical expression
+ Transformations
= High-performance code

Easy to capture GPU/CPU variants (different transformations)

Tiling, unrolling, prefetching, parallelization, data layout, ...
- Build quad-/octrees and FMM interaction lists for (optionally) sized sources/targets
- Compound sources
- Area queries
- Build tree-based geometry lookup tables
- Integration with (py)fmmlib
- Parallel via OpenCL
QBX FMM for layer potentials with close evaluation
High-order discretizations
Symbolic operators
Model: “Prepare, then do” (Allows optimization)
Parallel via OpenCL
Outline

1. Introduction
2. Developing QBX
3. Acceleration
4. Software
5. Results
BVP on a smooth domain
<table>
<thead>
<tr>
<th>BC</th>
<th>Side</th>
<th>k</th>
<th>p</th>
<th>( M = 70 )</th>
<th>( M = 105 )</th>
<th>( M = 130 )</th>
<th>EOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dir. int</td>
<td>1</td>
<td>1</td>
<td>1.7e-04 (16)</td>
<td>7.8e-05 (16)</td>
<td>8.0e-05 (16)</td>
<td>1.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>1.8e-06 (16)</td>
<td>4.4e-07 (16)</td>
<td>1.7e-07 (16)</td>
<td>3.8</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>5.7e-08 (16)</td>
<td>5.6e-09 (16)</td>
<td>3.7e-09 (16)</td>
<td>4.6</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>1</td>
<td>2.9e-02 (25)</td>
<td>1.3e-02 (25)</td>
<td>7.6e-03 (24)</td>
<td>2.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>8.1e-05 (25)</td>
<td>1.8e-05 (25)</td>
<td>6.1e-06 (25)</td>
<td>4.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>9.0e-07 (25)</td>
<td>9.7e-08 (25)</td>
<td>2.0e-08 (25)</td>
<td>6.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Neu. int</td>
<td>1</td>
<td>1</td>
<td>7.5e-02 (20)</td>
<td>5.4e-02 (20)</td>
<td>4.1e-02 (21)</td>
<td>0.9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>9.7e-04 (19)</td>
<td>3.2e-04 (19)</td>
<td>1.5e-04 (19)</td>
<td>3.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>1.2e-05 (18)</td>
<td>2.0e-06 (18)</td>
<td>6.6e-07 (18)</td>
<td>4.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>1</td>
<td>3.7e-01 (61)</td>
<td>3.2e-01 (61)</td>
<td>2.4e-01 (61)</td>
<td>0.7</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>2.8e-03 (60)</td>
<td>9.6e-04 (60)</td>
<td>4.4e-04 (60)</td>
<td>3.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>4.1e-05 (50)</td>
<td>6.5e-06 (50)</td>
<td>1.8e-06 (50)</td>
<td>4.9</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

BVP on a smooth domain

Andreas Klöckner

Fast Evaluation of Layer Potentials using QBX
Dirichlet problem on corner domain

![Graph of a corner domain](image)

Andreas Klöckner

Fast Evaluation of Layer Potentials using QBX
### Dirichlet problem on corner domain

<table>
<thead>
<tr>
<th>BC type</th>
<th>Side</th>
<th>k</th>
<th>p</th>
<th>$M = 80$</th>
<th>$M = 138$</th>
<th>EOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dirichlet</td>
<td>int</td>
<td>1</td>
<td>1</td>
<td>1.6e-03 (20)</td>
<td>2.1e-04 (21)</td>
<td>3.7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>1</td>
<td>5.9e-06 (20)</td>
<td>2.0e-07 (21)</td>
<td>6.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
<td>1</td>
<td>3.3e-08 (20)</td>
<td>3.3e-09 (21)</td>
<td>4.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6</td>
<td>1</td>
<td>6.9e-02 (38)</td>
<td>8.7e-03 (38)</td>
<td>3.8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>1</td>
<td>1.1e-04 (38)</td>
<td>3.8e-06 (38)</td>
<td>6.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
<td>1</td>
<td>1.0e-06 (38)</td>
<td>2.4e-08 (38)</td>
<td>6.9</td>
</tr>
<tr>
<td></td>
<td>ext</td>
<td>1</td>
<td>1</td>
<td>3.4e-04 (19)</td>
<td>5.2e-05 (19)</td>
<td>3.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>1</td>
<td>2.2e-06 (19)</td>
<td>8.2e-08 (19)</td>
<td>6.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
<td>1</td>
<td>1.3e-08 (19)</td>
<td>1.6e-09 (19)</td>
<td>3.9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6</td>
<td>1</td>
<td>1.6e-02 (33)</td>
<td>1.9e-03 (33)</td>
<td>3.9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>1</td>
<td>4.5e-05 (33)</td>
<td>1.3e-06 (33)</td>
<td>6.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
<td>1</td>
<td>1.4e-07 (33)</td>
<td>1.3e-08 (33)</td>
<td>4.4</td>
</tr>
</tbody>
</table>
Fast Evaluation of Layer Potentials using QBX
Andreas Klöckner

Fast Evaluation of Layer Potentials using QBX
For comparison: Double layer potential without QBX close evaluation
Layer potential FMM: Scaling behavior

Self-interaction: grid of circles

![Graph showing the relationship between execution time and number of source/target panels for self-interaction in a grid of circles. The x-axis represents the number of source/target panels ranging from 500 to 2000, and the y-axis represents execution time in seconds ranging from 0.2 to 1.0. The graph shows a decreasing trend followed by an increase with fluctuations.]
QBX: Why exciting?

Mathematically:
- General: Dimension, PDE, BC, kernel, surface discretization
- Fast enough for on-the-fly (non-stored) quadrature
- Benign conditioning
  (iterative methods $\rightarrow 10^{-15} \ll$ discr. error)
- Works easily for hypersingular kernels

Computationally:
- Little data, many flops: high arithmetic intensity
- Good match for hierarchical parallelism
- Locally homogeneous, batched work

http://www.cs.illinois.edu/~andreask/