ACM114 - Parallel Algorithms for Scientific Applications

Parallel implementation of the Boundary Element Method for the solution of the Laplace equation in unbounded domains

Carlos A. Pérez Arancibia

Computing & Mathematical Sciences
California Institute of Technology

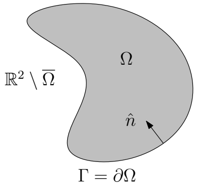
April 30, 2012

Setup of the problem

We want to solve the Laplace equation in a general unbounded domain:

Find $u : \mathbb{R}^d \setminus \overline{\Omega} \to \mathbb{R}$, $u \in H^1_{\text{loc}}(\mathbb{R}^d \setminus \overline{\Omega})$ $(d = 2, 3)$, such that:

$$(P) \begin{cases} -\Delta u = 0 & \text{in} \quad \mathbb{R}^d \setminus \overline{\Omega}, \\ \partial_{\boldsymbol{n}} u = q, \quad \text{or} \quad u = p & \text{on} \quad \Gamma, \\ + \text{ decay condition as } |\boldsymbol{x}| \to \infty \end{cases}$$


Ω
$\mathbb{R}^2 \setminus \overline{\Omega}$
$\hat{n}$
$\Gamma = \partial\Omega$

Applications

- Electrostatic and magnetostatic problems.
- Potential flow in fluid mechanics problems.
- Particle simulations (many body problem).

- The domain of definition of the solution is unbounded.
- Direct (naive) implementation of the Finite Differences or Finite Element method does not work in this case.
- We want to solve the problem for a general geometry.

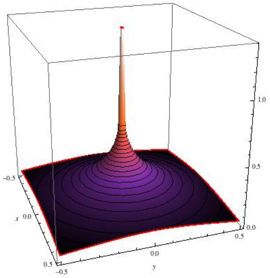
Boundary integral equation method

First we compute the Green's function of the problem:

$$G(\boldsymbol{x},\boldsymbol{y}) = \begin{cases} -\dfrac{1}{2\pi}\log(|\boldsymbol{x}-\boldsymbol{y}|) & \text{if} \quad d=2, \\ \dfrac{1}{4\pi|\boldsymbol{x}-\boldsymbol{y}|} & \text{if} \quad d=3, \end{cases}$$

solution of the following problem (in the sense of the distributions $\mathcal{D}'(\mathbb{R}^d)$):

$$-\Delta_{\boldsymbol{x}} G(\boldsymbol{x},\boldsymbol{y}) = \delta_{\boldsymbol{y}}(\boldsymbol{x}), \qquad \boldsymbol{x},\boldsymbol{y} \in \mathbb{R}^d.$$



Integral representation & boundary integral equation

Using the second Green's identity we get:

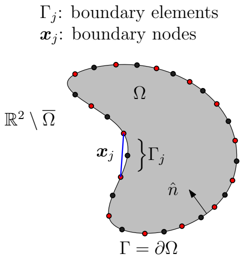
$$\text{BIR:} \quad u(\boldsymbol{x}) = \int_\Gamma \left\{ G(\boldsymbol{x},\boldsymbol{y}) q(\boldsymbol{y}) - \frac{\partial G}{\partial n_{\boldsymbol{y}}}(\boldsymbol{x},\boldsymbol{y}) p(\boldsymbol{y}) \right\} \mathrm{d}\sigma_{\boldsymbol{y}}, \quad \boldsymbol{x} \in \mathbb{R}^d \setminus \overline{\Omega},$$

$$\text{BIE:} \quad \frac{p(\boldsymbol{x})}{2} = \int_\Gamma \left\{ G(\boldsymbol{x},\boldsymbol{y}) q(\boldsymbol{y}) - \frac{\partial G}{\partial n_{\boldsymbol{y}}}(\boldsymbol{x},\boldsymbol{y}) p(\boldsymbol{y}) \right\} \mathrm{d}\sigma_{\boldsymbol{y}}, \quad \boldsymbol{x} \in \Gamma.$$

Boundary element method

We discretize the boundary integral equation (BIE) by seeking a solution of the form

$$p(\boldsymbol{x}) \approx \sum_{j=1}^{N} p_j \mathbf{1}_{\Gamma_j}(\boldsymbol{x})$$


Γ_j: boundary elements
$\boldsymbol{x}_j$: boundary nodes
Ω
$\mathbb{R}^2 \setminus \overline{\Omega}$
$\boldsymbol{x}_j$
$\}\Gamma_j$
$\hat{n}$
$\Gamma = \partial\Omega$

which leads to the following set of equations:

$$\frac{p_i}{2} = \sum_{j=1}^{N} \{g_{ij} q_j - f_{ij} p_j\}, \quad \forall i = 1, \ldots, N$$

where

$$g_{ij} = \int_{\Gamma_j} G(\boldsymbol{x}_i, \boldsymbol{y})\, \mathrm{d}\sigma_{\boldsymbol{y}}, \quad f_{ij} = \int_{\Gamma_j} \frac{\partial G}{\partial n_{\boldsymbol{y}}}(\boldsymbol{x}_i, \boldsymbol{y})\, \mathrm{d}\sigma_{\boldsymbol{y}}.$$

The approximate values of the solution or its normal derivative can be computed by solving the linear system

$$\left(\frac{\mathbf{I}}{2} + \mathbf{F}\right) \mathbf{p} = \mathbf{G}\, \mathbf{q}.$$

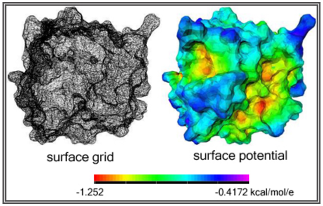
In other words, we have the linear system:

$$\begin{array}{lrcll} \text{Dirichlet:} & \mathbf{A}\,\mathbf{q} & = & \mathbf{b}, & \mathbf{A} = \mathbf{G}, & \mathbf{b} = (\mathbf{I}/2 + \mathbf{F})\,\mathbf{p} \\ \text{Neumann:} & \mathbf{A}\,\mathbf{p} & = & \mathbf{b}, & \mathbf{A} = (\mathbf{I}/2 + \mathbf{F}), & \mathbf{b} = \mathbf{G}\,\mathbf{q} \end{array}$$

Parallelization

Why parallelize?

- High-resolution meshes or large number of objects (e.g. galaxies or molecules) requires a large number of elements.
- The computation of the matrix entries requires at least $\mathcal{O}(N^2)$ operations.
- Unlike FEM or FDM, the system matrix arising in BEM is nonsymetric and dense, so standard fast algorithms (e.g. cyclic reduction methods) do not work in this case.
- For large N the numerical solution of the resulting linear system may be infeasible expensive even for Krylov subspace methods, which require only $\mathcal{O}(N^2)$ operations to converge for a general well-conditioned matrix.
- The evaluation of the numerical solution at a single arbitrary point outside the boundary requires $\mathcal{O}(N^2)$ as well.

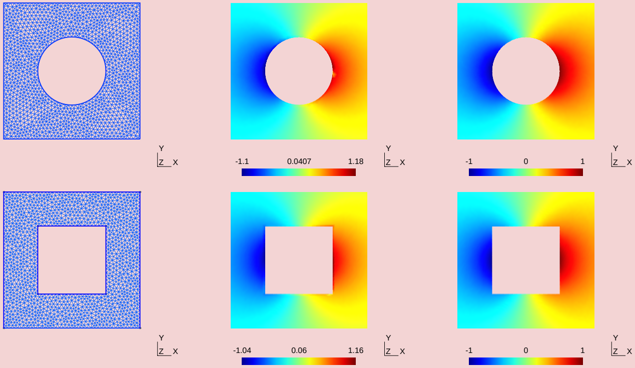


Parallel implementation

1. The mesh data is made available to all p processors.
2. Each processor computes and store $\delta w = N/p$ rows of $\mathbf{F}$ and $\mathbf{G}$. It allows to perform matrix-vector multiplications in parallel by computing the dot product between the matrix rows and the vector. This operation takes $\mathcal{O}(N^2/p)$ time units.
3. The boundary condition vector is distributed to all the processors and then $\mathbf{b}$ is computed. Each processor computes δw entries of $\mathbf{b}$ and send them to the master process. This operation requires $\mathcal{O}(N^2/p)$ time units.
4. The linear system is solved using the GMRES method. It only requires matrix-vector products of $\mathbf{A}$. Each GMRES iteration requires $\mathcal{O}(N^2/p)$ time units. If $\mathbf{A}$ is well-conditioned, the solution of the linear system takes $\mathcal{O}(N^2/p)$ time units.
5. The integral representation is evaluated at N_p points. Each processor computes u at $\delta w_p = N_p/p$ different points. Subsequently, these values are sent to the master process where a vector stores u for visualization purposes.

Benchmark problem

- We consider the function $p(r, \theta) = \cos\theta / r$, which is analytical and satisfies $-\Delta p = 0$ for all $\boldsymbol{x} = r(\cos\theta, \sin\theta) \in \mathbb{R}^2 \setminus \{(0,0)\}$.
- We solve the Laplace equation for the condition $u = p$ on $\Gamma = \partial\Omega$. As result we recover $q = \partial u / \partial n$ on the boundary and the values of u at points $\boldsymbol{x} \in \mathbb{R}^2 \setminus \overline{\Omega}$ by using the integral representation formula.
- For $N = 60$ the error obtained for the normal derivative is $\max_{\boldsymbol{x}\in\Gamma} |q - \partial u/\partial n| =$ 1.94E-03.


Y
Z X
-1.1
0.0407
1.18
-1
0
1
-1.04
0.06
1.16