Contents lists available at ScienceDirect

Computer Physics Communications

journal homepage: www.elsevier.com/locate/cpc

A GPU accelerated and error-controlled solver for the unbounded Poisson equation in three dimensions[☆]

Lukas Exl [*]

Fak. Mathematik, University of Vienna, Oskar-Morgenstern-Platz 1, 1090, Vienna, Austria
Physics of Functional Materials, University of Vienna, 1090 Vienna, Austria

ARTICLE INFO

Article history:
Received 6 December 2016
Received in revised form 26 April 2017
Accepted 7 August 2017
Available online 24 August 2017

Keywords:
Convolution via fast Fourier transform (FFT)
GPU computing
Free space Coulomb/dipole–dipole potential
Separable Gaussian-sum (GS) approximation

ABSTRACT

An efficient solver for the three dimensional free-space Poisson equation is presented. The underlying numerical method is based on finite Fourier series approximation. While the error of all involved approximations can be fully controlled, the overall computation error is driven by the convergence of the finite Fourier series of the density. For smooth and fast-decaying densities the proposed method will be spectrally accurate. The method scales with $\mathcal{O}(N \log N)$ operations, where N is the total number of discretization points in the Cartesian grid. The majority of the computational costs come from fast Fourier transforms (FFT), which makes it ideal for GPU computation. Several numerical computations on CPU and GPU validate the method and show efficiency and convergence behavior. Tests are performed using the Vienna Scientific Cluster 3 (VSC3). A free MATLAB implementation for CPU and GPU is provided to the interested community.

Program summary

Program Title: `GSPoisson3d`
Program Files doi: http://dx.doi.org/10.17632/xh6d47sxx8.1
Licensing provisions: MIT
Programming language: MATLAB R2015b
Nature of problem: Efficient and accurate computation of the unbounded Poisson equation in three dimensions.
Solution method: Fourier based approach with Gaussian-sum approximation of the singular convolution kernel and near field correction — both utilizing FFT.
Additional comments: Incorporated GPU acceleration via MATLAB's GPU `fft` implementation.

1. Introduction

The purpose of this paper is to provide the interested reader with a MATLAB implementation of an efficient and mathematically analyzed method [1] for solving the free-space/unbounded Poisson equation. More precisely, the method presented in this paper solves

$$-\Delta u(\mathbf{x}) = \rho(\mathbf{x}), \quad \mathbf{x} \in \mathbb{R}^3, \quad \lim_{|\mathbf{x}|\to\infty} |u(\mathbf{x})| = 0, \tag{1}$$

via the well-known representation of the solution to (1) as the convolution of the density ρ with the free-space Green's function $U(\mathbf{x}) = \frac{1}{4\pi}\frac{1}{|\mathbf{x}|}$

$$u(\mathbf{x}) = (U * \rho)(\mathbf{x}) = \int_{\mathbb{R}^3} U(\mathbf{x}-\mathbf{y})\rho(\mathbf{y})d\mathbf{y}, \quad \mathbf{x} \in \mathbb{R}^3. \tag{2}$$

The problem (1) is fundamental in many fields of physics, e.g. quantum chemistry [2–7], particle physics [8,9] or astrophysics [10]. Therefore, the provided implementation might serve as improvement of existing simulation codes that use the high-level computing environment MATLAB. However, the provided code could also be understood as an easily readable open source prototype, ready for translation to other different programming languages.

The proposed method relies on a smooth Gaussian-sum approximation of the singular free-space Green's function and a correction step. A related method was described in [4], which also uses optimized exponential sums to approximate the kernel separably but implements no correction. Although this method already shows accuracy in the range of single precision and slightly below, it uses a different and larger sum of Gaussian functions

[☆] This paper and its associated computer program are available via the Computer Physics Communication homepage on ScienceDirect (http://www.sciencedirect.com/science/journal/00104655).

[*] Correspondence to: Fak. Mathematik, University of Vienna, Oskar-Morgenstern-Platz 1, 1090, Vienna, Austria.
E-mail address: lukas.exl@univie.ac.at.

http://dx.doi.org/10.1016/j.cpc.2017.08.014
0010-4655/© 2017 Elsevier B.V. All rights reserved.

and does not achieve similar high accuracy because of lack of a correction step. The exceptional high accuracy and efficiency of the method presented in the following arises from two novel aspects, whose consideration does not lead to any computational disadvantage over previously introduced approaches. Firstly, the tensor product structure of the Gaussian-sum approximation allows utilization of highly accurate one-dimensional quadrature in a setup phase. This is followed by convolution via fast Fourier transforms with doubled sizes owing to zero padding. Secondly, the near zone correction supplements the smooth kernel approximation via Taylor expansion of the density. This step is also performed by Fast Fourier transform in quasi-linear time. Overall, the proposed method scales quasi-linearly, where the computational costs are mainly due to Fast Fourier transforms.

In Section 2 the method is described mathematically, followed by a section about computational aspects and approximation errors (see Section 3). Section 4 describes the usage of the implementation by means of a test example. Validation of the implementation and tests for computational efficiency (see Section 5) show practical applicability. Test runs are performed on the Vienna Scientific Cluster 3 (VSC3) both on CPU nodes and Tesla GPU devices. For background information about usage, efficiency and application of GPU-based fast Fourier transform the interested reader is referred to the recent publication [11].

2. Method description

The method of this paper is in the class of Ewald type methods [13,3,12]. Those approaches split the singular convolution kernel U into a smooth long-range part U_s and a singular short-range correction U_c. The smooth part of the convolution can then be treated with help of the convolution theorem, i.e., $U * \rho = \mathcal{F}^{-1}(\mathcal{F}(U_s) \cdot \mathcal{F}(\rho))$. This is usually done on an equispaced Cartesian grid with the help of the quasi linearly scaling fast Fourier transform. Here, the smoothness usually leads to fast converging Fourier series, which make the discrete approximation accurate even on coarser grids. However, the correction U_c still contains a singularity but is also localized, hence, can be treated with a direct summation approach. While a direct evaluation of the convolution (1) would scale with $\mathcal{O}(N^2)$ on a Cartesian grid with a total number of N grid points, the original Ewald method [12] and parameter tuned variations of it scale with $\mathcal{O}(N^{3/2})$ operations. The method described here scales with $\mathcal{O}(N \log N)$ operations. This is achieved by FFT for both parts, the smooth convolution and the correction. The smooth kernel consists of a product of one-dimensional exponential functions (Gaussian-sum), which allows the usage of highly accurate one-dimensional adaptive quadrature for computation of the interaction kernel in Fourier space. Taylor expansion of the density in the near-zone allows to treat the correction by analytical integration, where involved derivatives are computed by FFT as well. The method is efficient and mathematically proven to yield full control over the maximum computation error [1].

We now give a brief description of the method, where we also emphasize novel aspects relevant to the implementation. A detailed mathematical description of the method including error analysis was recently published by the author [1]. We adapt it here for the case of general rectangular computational domains.

The computational box coincides with the domain of target points $\mathbf{x}$, where the potential u is computed. The smooth density ρ is assumed to vanish (up to double precision) outside the computational box, so it is expected to be fast decaying. Our method makes use of a finite Fourier series approximation of the density, which is assumed to be fast converging due to the smoothness and compact support of ρ. The analysis in [1] assumes the computational domain to be the unit square box $\mathbf{B}_1 := [-1, 1]^3$. To generalize the method's framework for the user's convenience, assume the density ρ to be compactly supported in the general rectangular box $\mathbf{B} := [a_1, b_1] \times [a_2, b_2] \times [a_3, b_3]$. We will first derive a 'standardized form' of the problem, which allows us to treat the key-approximations of the method independently of the concrete choice of the computational box. Let $\mathbf{c}$ be the center of $\mathbf{B}$ such that $\widetilde{\mathbf{B}} := \mathbf{B} - \mathbf{c}$ is a rectangular box centered at the origin. As a consequence of the compact support of the density, the convolution integral (2) is actually over the domain $\mathbf{B}$. Hence, we can write

$$u(\mathbf{x}) = \int_{\widetilde{\mathbf{B}}} U(\widetilde{\mathbf{x}} - \widetilde{\mathbf{y}})\rho(\widetilde{\mathbf{y}} + \mathbf{c})d\widetilde{\mathbf{y}}, \quad \mathbf{x} = \widetilde{\mathbf{x}} + \mathbf{c} \in \mathbf{B}. \tag{3}$$

Now we define $\lambda := \max_{q=1,2,3}\{\frac{b_q - a_q}{2}\}$ and $\mathbf{B}_{1,\lambda} := \frac{1}{\lambda}\widetilde{\mathbf{B}} \subseteq \mathbf{B}_1$. We get

$$u(\mathbf{x}) = \lambda^2 \int_{\mathbf{B}_{1,\lambda}} U(\mathbf{x}' - \mathbf{y}')\rho(\lambda\mathbf{y}' + \mathbf{c})d\mathbf{y}', \quad \mathbf{x} = \lambda\mathbf{x}' + \mathbf{c} \in \mathbf{B}. \tag{4}$$

Changing variables and extending the domain of integration to $2 \cdot \mathbf{B}_{1,\lambda} \supseteq \mathbf{x}' + \mathbf{B}_{1,\lambda},\ \mathbf{x}' \in \mathbf{B}_{1,\lambda}$ leads to

$$u(\mathbf{x}) = \lambda^2 \int_{2\cdot\mathbf{B}_{1,\lambda}} U(\mathbf{y})\rho_{\lambda;\mathbf{c}}(\mathbf{x}' - \mathbf{y})d\mathbf{y}, \quad \mathbf{x} = \lambda\mathbf{x}' + \mathbf{c} \in \mathbf{B}, \tag{5}$$

where $\rho_{\lambda;\mathbf{c}}(\mathbf{x}) := \rho(\lambda\mathbf{x} + \mathbf{c})$ with support in $\mathbf{B}_{1,\lambda}$. The extension of the integration domain in (5) does not change the value of the integral but it makes the domain independent of the target points. This aspect makes it possible to calculate a large portion of the computational task in a precomputation phase, compare with (7).

The key idea of the method is to approximate the singular kernel $U(\mathbf{x}) = \frac{1}{4\pi}\frac{1}{|\mathbf{x}|}$ with a Gaussian-sum in a region contained in the integration domain $2 \cdot \mathbf{B}_{1,\lambda}$ but excluding a δ-ball around the origin (where U is singular). The latter step is compensated by a near zone correction. More precisely, for $\delta \in (0, \min_{q=1,2,3}\{\frac{b_q - a_q}{2\lambda}\})$ and $\mathbf{x} = \lambda\mathbf{x}' + \mathbf{c} \in \mathbf{B}$ we get

$$u(\mathbf{x}) \approx \lambda^2 \left(\int_{2\cdot\mathbf{B}_{1,\lambda}} U_{GS}(\mathbf{y})\rho_{\lambda;\mathbf{c}}(\mathbf{x}' - \mathbf{y})d\mathbf{y} + \int_{\mathbf{B}_\delta} (U - U_{GS})(\mathbf{y})\rho_{\lambda;\mathbf{c}}(\mathbf{x}' - \mathbf{y})d\mathbf{y} \right) =: \lambda^2\big(I_1(\mathbf{x}) + I_\delta(\mathbf{x})\big), \tag{6}$$

where $U_{GS}(\mathbf{y}) = \sum_{j=0}^{S} w_j e^{-\tau_j^2|\mathbf{y}|^2} = \sum_{j=0}^{S} w_j \prod_{q=1}^{3} e^{-\tau_j^2 y_q^2} \approx U(\mathbf{y})$, $|\mathbf{y}| \in [\delta, 2]$ is a Gaussian-sum (GS) approximation realized by sinc-quadrature [14,15,1]. The integrand in $I_1(\mathbf{x})$ is smooth and its convolution kernel is separable (product of 1d functions). Hence, it can be treated efficiently by a Fourier based approach. More precisely, it is computed by the inverse Fourier transform of the product of the Fourier transform of the density with the G-tensor

$$G_{\mathbf{k}} = \sum_{j=0}^{S} w_j G_{\mathbf{k}}^j$$
$$\text{with } G_{\mathbf{k}}^j = G_{k_1k_2k_3}^l = \prod_{q=1}^{3} \int_0^2 2l_q\, e^{-(\tau_j l_q)^2 y_q^2} \cos(\frac{\pi}{2}k_q y_q)\, dy_q, \tag{7}$$

where $\mathbf{B}_{1,\lambda} = [-l_1, l_1] \times [-l_2, l_2] \times [-l_3, l_3]$. The G-tensor can be computed accurately by one-dimensional adaptive Gauss–Kronrod quadrature in a setup phase. The two Fourier transforms in the (run-time) computation of I_1 are efficiently implemented via the FFT with zero-padding, which increases the effort by a factor of eight.

The correction integral I_δ is calculated by inserting the third order Taylor polynomial of the shifted density $\rho_{\lambda;\mathbf{c};\mathbf{x}'}(\mathbf{y}) := \rho_{\lambda;\mathbf{c}}(\mathbf{x}' - \mathbf{y})$ around $\mathbf{0}$, followed by analytical integration in spherical coordinates. The contributions of odd derivatives in the Taylor