Available online at www.sciencedirect.com
ScienceDirect
SoftwareX 5 (2016) 178–182
SoftwareX
www.elsevier.com/locate/softx

A GPU code for analytic continuation through a sampling method

Johan Nordström[a,*], Johan Schött[a], Inka L.M. Locht[a,b], Igor Di Marco[a]

[a] Department of Physics and Astronomy, Uppsala University, Box 516, SE-75120 Uppsala, Sweden
[b] Institute of Molecules and Materials, Radboud University of Nijmegen, Heyendaalseweg 135, 6525 AJ Nijmegen, The Netherlands

Received 9 February 2016; received in revised form 22 June 2016; accepted 23 August 2016

Abstract

We here present a code for performing analytic continuation of fermionic Green's functions and self-energies as well as bosonic susceptibilities on a graphics processing unit (GPU). The code is based on the sampling method introduced by Mishchenko et al. (2000), and is written for the widely used CUDA platform from NVidia. Detailed scaling tests are presented, for two different GPUs, in order to highlight the advantages of this code with respect to standard CPU computations. Finally, as an example of possible applications, we provide the analytic continuation of model Gaussian functions, as well as more realistic test cases from many-body physics.

Keywords: GPU; Analytic continuation; Parallelization; Green's function

Code metadata

Current code version	Version 1.0
Permanent link to code/repository used of this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-16-00025
Legal Code License	GPL-3.0
Code versioning system used	none
Software code languages, tools, and services used	C++ with CUDA
Compilation requirements, operating environments & dependencies	Linux, CUDA compatible NVIDIA GPU; CUDA Toolkit 7 with thrust library
If available Link to developer documentation/manual	Userguide included in the code package
Support email for questions	cjs.nordstrom@gmail.com

1. Introduction

The last two decades have seen a drastic increase of computational studies in solid state physics. These not only comprehend works based on density functional theory [1], but also more involved works based on many-body theory [2]. Solving the many-body problem usually involves the calculation of Green's functions and self-energies. At finite temperature, it becomes more convenient to perform this calculation at imaginary times, or equivalently at imaginary energies. Although this makes it possible to perform more efficient simulations, the physical observables one is interested in are still defined at real times, or equivalently at real energies. Hence, one needs to perform analytic continuation of Green's functions and self-energies in the complex plane. Unfortunately, this results in an ill-posed problem, whose solution may be difficult to obtain without *a priori* knowledge on the function itself. For these reasons, a variety of numerical methods for performing the analytic continuation exist. The most celebrated methods include the maximum entropy method [3–10], the Padé approximant method [11–14], the Singular Value Decomposition [15,10], the non-negative Tikhonov method [16], the non-negative least-square method [17], stochastic regularization methods [18] and sampling methods [19–22].

In this article we focus on the approach proposed by Mishchenko et al. in Ref. [19], where the analytic continuation

* Corresponding author.
E-mail address: cjs.nordstrom@gmail.com (J. Nordström).

http://dx.doi.org/10.1016/j.softx.2016.08.003

is approximated with a sum of rectangles to be determined via stochastic sampling. Although this method has raised high expectations in the scientific community, its applicability has been so far limited by the high computational cost. This requires that one needs to perform the analytic continuation on supercomputers facilities, which are not always accessible. Moreover, one has often to experience long queuing times, which makes the data analysis particularly inconvenient. It would be desirable to be able to work directly on a common laptop. To fulfill this need, we present here a code for the analytic continuation through Mishchenko's method for graphics processing units (GPUs). At the moment the code is applicable to fermionic Green's functions and self-energies, as well as to bosonic susceptibilities. For typical problems it is possible to perform several analytic continuations in a few minutes on a common laptop, which is going to significantly extend the number of researchers interested in Mishchenko's approach. Moreover, some of the presented numerical routines may find applicability for other problems requiring a stochastic sampling.

2. Problems and background

This program provides an approximate solution to the ill-posed Fredholm integral equation

$$G(x) = \int_a^b K(x,\omega)\rho(\omega)d\omega \tag{1}$$

where G, K are known functions and ρ is unknown. For our purposes $G(x)$ can be a fermionic Green's function, a fermionic self-energy or a bosonic susceptibility with the symmetry that its spectral function is odd. For the first two, x is either the imaginary time τ or the imaginary fermionic (Matsubara) frequency $i\omega_n$. For a bosonic function x can only be the imaginary bosonic frequency $i\omega_n$ (extensions to τ are planned in the future). The function K in Eq. (1) is known as the kernel, and for the three possibilities above can take the form:

$$K_{\mathrm{f}}(\tau,\omega) = \frac{\exp(-\tau\omega)}{1+\exp(-\beta\omega)} \qquad K_{\mathrm{f}}(i\omega_n,\omega) = \frac{1}{i\omega_n-\omega}$$
$$K_{\mathrm{b}}(i\omega_n,\omega) = \frac{\omega^2}{\omega_n^2+\omega^2} \tag{2}$$

where the subscripts f and b stay for fermionic and bosonic respectively. The wanted unknown function $\rho(\omega)$ is the spectral function, defined as a function of real energies ω.

2.1. Mishchenko's stochastic optimization method

To obtain a solution $\rho(\omega)$ to Eq. (1), A.S. Mishchenko et al. [19] have developed an approach based on the Monte Carlo method. The main idea is to approximate the true $\rho(\omega)$ with $\tilde{\rho}(\omega)$, which is a sum of R rectangles, then calculate $\tilde{G}(\tau)$ for that particular $\tilde{\rho}(\omega)$ and compare the result to the known values of $G(\tau)$ to get an indication of how adequate the guess $\tilde{\rho}(\omega)$ was. The rectangles are finally updated in a Monte Carlo fashion, as described below. More in detail, $\tilde{\rho}(\omega)$ can be written as

$$\tilde{\rho}(\omega) = \sum_{i=1}^{R} \chi_{\{P_i\}}(\omega) \tag{3}$$

where the sum runs over R rectangles, which are defined as

$$\chi_{\{P_i\}}(\omega) = \begin{cases} h_i, & \omega \in [c_i - w_i/2, c_i + w_i/2] \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

Here $\{P_i\} = \{h_i, w_i, c_i\}$ represents width, height and position (referred to the middle point). A set of rectangles is called a configuration and corresponds to a given $\tilde{\rho}(\omega)$. The latter can be transformed into the corresponding $\tilde{G}(x)$ using Eq. (1). Given the fact that the rectangle function $\chi_{\{P_i\}}(\omega)$ is constant with height h_i, inserting Eq. (3) into Eq. (1) gives

$$\tilde{G}(x) = \sum_{i=1}^{R} h_i \int_{c_i-w_i/2}^{c_i+w_i/2} K(x,\omega)d\omega. \tag{5}$$

The integral within the sum can either be solved analytically or calculated numerically depending on the kernel K. In a practical situation the function to continue is known at a finite set of N points. To express the deviation between $\tilde{G}(x)$ obtained from configuration $\tilde{\rho}(\omega)$ and the true $G(x)$ that is known at N points, one can use the following deviation measure [19]:

$$D[\tilde{\rho}(\omega)] = \sum_{j=1}^{N} \left| \frac{G(x_j) - \tilde{G}(x_j)}{G(x_j)} \right|. \tag{6}$$

Minimizing Eq. (6) numerically is an ill-conditioned problem but by generating M independent solutions and taking the average

$$\rho(\omega) = \frac{1}{M}\sum_{j=1}^{M} \tilde{\rho}_j(\omega) \tag{7}$$

the noise of the independent solutions averages out [19].

The implemented algorithm starts with an initial configuration and iteratively applies random changes to the configuration to reduce the deviation $D[\tilde{\rho}(\omega)]$ in Eq. (6). The full method can be separated into elementary, local and global updates of the configurations:

Elementary update: A single alteration C of a configuration such as e.g. changing the width of a rectangle or shifting its center.

Local update: A series of E consecutive elementary updates:

$$C(0) \to C(1) \to \cdots \to C(r) \to C(r+1) \to \cdots \to C(E). \tag{8}$$

Each elementary update is either accepted or discarded according to a probability function. The result of the local update is the configuration of the series that has the lowest deviation $D[\tilde{\rho}(\omega)]$.

Global update: A series of L local updates, each including E elementary updates, starting from a randomly

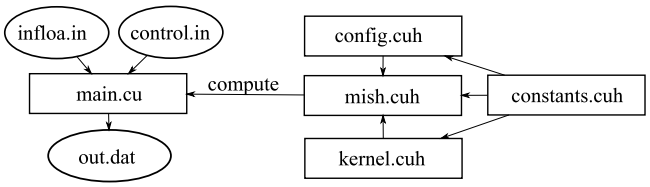


Fig. 1. Overview of the software structure and interactions. *main.cu* is an example program which uses the library by reading two input files and produces *out.dat* which contains the output spectral function.

generated initial configuration. The global update results in one independent solution.

The average of all independent solutions of the global updates, as in Eq. (7), provides the final approximation to $\rho(\omega)$.

3. Software description

3.1. Software architecture

The developed software is a GPU accelerated C++ code for the stochastic optimization method outlined above. All GPU related operations are based on CUDA, which is a parallel computing platform and programming model for NVIDIA graphic cards.

The software structure and interactions are shown in Fig. 1. The main computational library is accessed through the function *compute* which is defined in *mish.cuh*. This function takes three input arguments, i.e. an input structure and two float arrays, which contain the output spectral function ρ and its argument ω. The details of the input structure are described in the source code.

3.2. Software functionalities

The standard and recommended way of using the present software is through the program *main.cu*, located in the source folder. This program reads two input files, *control.in* and *infloa.in*. The former specifies both algorithm-specific and problem-specific parameters, while the latter contains the input function for a finite number of points (τ or $i\omega_n$). If one intends to perform analytical continuation of a self-energy, first the Hartree–Fock term should be removed, then the data should be normalized to get analogous asymptotics as for a Green's function. After the continuation, the self-energy spectrum has to be renormalized with the same factor as the input. For a bosonic function the input Matsubara data has to be divided by $G(i\omega_n = 0)$, while the output function has to be multiplied with $-\omega G(i\omega_n = 0)/2$. The file *infloa.in* requires input data in the column format x, Re$[G(x)]$, Im$[G(x)]$. For real valued input functions, the imaginary part is simply left out. The file *out.dat* contains the resulting spectral density function from Eq. (7) in the column format ω, $\rho(\omega)$. The output is space separated and can be viewed directly by e.g. gnuplot.

3.3. Implementation details

The parallelization of the algorithm is done in two layers. Firstly, each particular solution of Eq. (7) can be generated independently. This is known as an *embarrassingly parallel* task. The second layer is parallelization of selected tasks within the computation of one particular solution. This specifically includes the computation of Eq. (5). The sum for each x can be computed independently. Eq. (6) is also done in parallel as well as various memory transfer related tasks.

The main CUDA kernel, which does the majority of the computations, is executed using a one dimensional CUDA grid with size equal to the number of global updates. The CUDA block size is set to be equal to the number of input Green's function points. This means that the grid and block size are fixed depending on the input parameters. A drawback with this is that the occupancy of the kernel might be low if the number of input points is low. This also means that there exist a maximum amount of input points which is limited by the hardware, typically 512 or 1024 input points. However, in our experience, this is not an issue in real world cases.

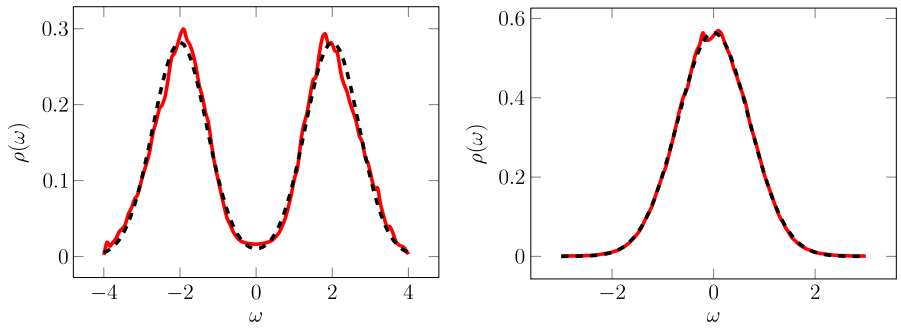
The significant data is stored in global memory on the GPU in single precision and the total GPU memory usage can exceed hundreds of megabytes. The algorithm implementation is heavy on memory load/store operations on the GPU which is the current main performance bottleneck.

4. Illustrative examples

The software is tested through an accurate comparison with existing CPU implementations of the stochastic optimization method [19], and a very good agreement is found. Illustrative examples can be provided for the analytic continuation of known functions. This implies first converting a known spectral function to a Green's function through Eq. (1) and then performing the analytic continuation through our software.

Fig. 2 shows a comparison between the analytic continuation obtained by means of our code versus the exact result for two simple test-cases consisting of (fermionic) Gaussian functions. Although these tests involve only symmetric functions with a limited number of peaks (one or two), the provided software reconstructs their shapes very well in both cases. A much more demanding test is presented in Fig. 3(a), for the atomic-like self-energy of a Sm atom in a cluster containing seven Sm atoms, as described in Ref. [14]. This is the most demanding test for the analytic continuation, since it involves resolving several narrow peaks at short distances from each other. Our software, as well as CPU versions of Mishchenko's method, captures only the two main peaks close to zero energy but fail to resolve the high energy peaks. One should not think that these issues are proper of the Mishchenko's method alone. Other techniques as well have severe problems in reproducing the high energy peaks, unless very high precision data are provided, which are not always accessible. In Fig. 3(b) we instead present an example of analytical continuation of a bosonic susceptibility for a non-interacting electron system on a cubic lattice in random phase approximation, as described in Ref. [23].

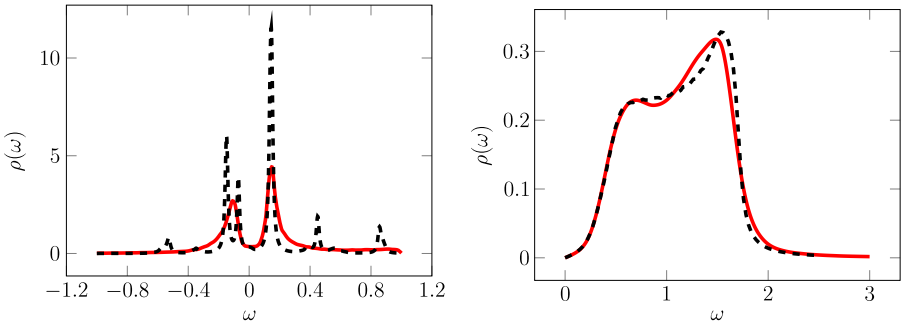
Our software will be mostly applied to perform the analytic continuation of data plagued by numerical noise, e.g. from quantum Monte-Carlo simulations [24]. Illustrative examples for the Hubbard model on an infinite-dimensional Bethe lattice



(a) 128 Green's points, 512 Global updates, 80 local updates, 80 elementary updates.

(b) 128 Green's points, 512 Global updates, 80 local updates, 80 elementary updates.

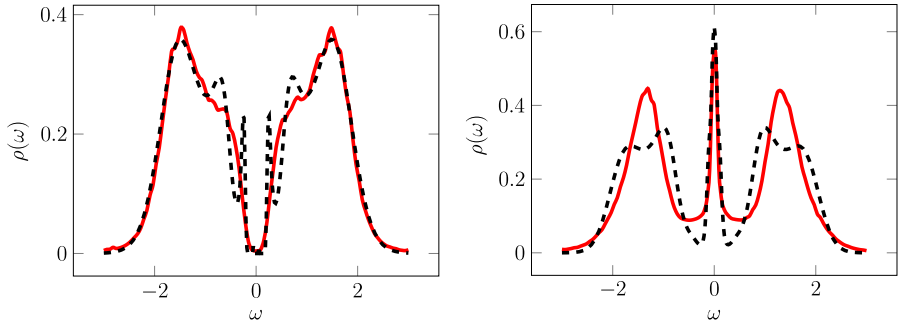
Fig. 2. Two different Green's functions consisting of Gaussians. The dashed black curve is the exact solution and the red curve is the approximated solution obtained by the presented software. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



(a) Sm self-energy.

(b) Bosonic susceptibility.

Fig. 3. Imaginary part of the atomic-like self-energy of a Sm atom in a cluster and of a bosonic susceptibility for a non-interacting electron system on a cubic lattice. The generated functions are shown in red, while the expected functions are in dashed black. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



(a) Insulator.

(b) Metal.

Fig. 4. Spectral functions for the Hubbard model (see main text). In panel a (b) the insulating (metallic) phase is shown. The solid red line indicates the solution obtained with our code, while the dashed black line corresponds to results obtained with the maximum entropy method [3–5]. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

at half-filling for $U/W = 2.5$ and $\beta = 150$ are reported in Fig. 4. Here U is the strength of the local Coulomb repulsion, W is half the bandwidth for the non-interacting system, and β is the inverse temperature. For these parameters the fermionic Green's function has two solutions; an insulating one and a metallic one. Continuations for both solutions are reported in Fig. 4. Given that the exact analytic form of the solution is not known, we compare our results to the maximum entropy method [3–5]. The agreement is in general good. The maximum entropy method seems to have a slightly higher resolving power, but requires a priori information on the shape of the function, while the stochastic optimization method is completely unbiased. In fact, one of the most convenient applications of the stochastic optimization method is to provide an initial estimate of the true spectral function, which can then be used as an input for other techniques, e.g. as a model function for the maximum entropy method.

4.1. Performance

To our knowledge, the existing implementations of the stochastic optimization method by Mishchenko et al. are not very well optimized and thus, not suitable for speedup mea-

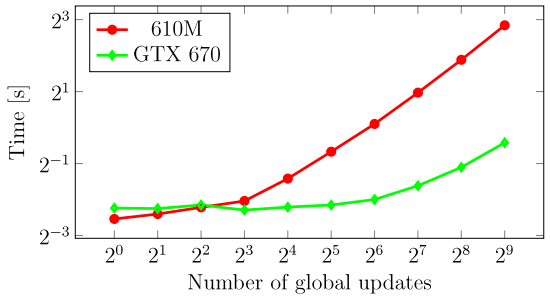


Fig. 5. Logarithmic plot (both x- and y-axis) of time versus number of global updates. For the green curve (diamonds) we used a GTX 670, and for the red curve (circles) a 610M GPU. *[128 threads per block, 40 local updates, 40 elementary updates]* (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

surements of GPU versus CPU. One simple metric to measure the parallel efficiency is by looking at the total execution time while varying the global updates (number of particular solutions). The total amount of computations increases linearly with each added global update. Since we scale the number of GPU cores used with the number of global updates, a perfectly parallel execution would require a constant runtime, regardless of the number of global updates. This feature is tested for two different GPUs, and the results are shown in Fig. 5. The GTX 670 is a faster and more efficient GPU compared with the 610M. Fig. 5 shows that the GTX 670 scales better when increasing the number of global updates. This indicates that the implementation scales very well when increasing the parallel capacity.

5. Conclusions

We have presented a GPU implementation of the stochastic optimization method by Mishchenko et al. [19] to perform the analytic continuation of Green's functions. Our software makes it possible to obtain results with sufficient accuracy (i.e. a stochastic sampling which is large enough) on a common laptop, without requiring the access to a supercomputer. For a typical problem with 64 input points, in only 25 s our implementation can do about 200 elementary updates for 200 local updates and a total of 128 global updates, which corresponds to a total of 5 120 000 elementary updates. We are confident that this code can contribute to increase the usage of the stochastic optimization method among scientists working in many-body theory.

Acknowledgments

We thank Andrea Droghetti for stimulating discussions on the stochastic optimization method. This work was sponsored by the Swedish Research Council (VR), the Swedish strategic research programme eSSENCE and the Knut and Alice Wallenberg foundation (KAW, grants 2013.0020 and 2012.0031). The computations with the CPU version of the code were performed on resources provided by the Swedish National Infrastructure for Computing (SNIC).

References

[1] Martin RM. Electronic structure: basic theory and practical methods. Cambridge: Cambridge University Press; 2004.
[2] Mahan GD. Many-particle physics. New York and London: Plenum Press; 1993.
[3] Silver RN, Sivia DS, Gubernatis JE. Maximum-entropy method for analytic continuation of quantum monte carlo data. Phys Rev B 1990; 41:2380–9.
[4] Jarrell M, Gubernatis J. Bayesian inference and the analytic continuation of imaginary-time quantum monte carlo data. Phys Rep 1996;269(3): 133–95.
[5] Bryan R. Maximum entropy analysis of oversampled data problems. European Biophys J 1990;18(3):165–74.
[6] Gubernatis JE, Jarrell M, Silver RN, Sivia DS. Quantum monte carlo simulations and maximum entropy: Dynamics from imaginary-time data. Phys Rev B 1991;44:6011–29.
[7] Fuchs S, Pruschke T, Jarrell M. Analytic continuation of quantum monte carlo data by stochastic analytical inference. Phys Rev E 2010;81:056701.
[8] Dirks A, Werner P, Jarrell M, Pruschke T. Continuous-time quantum monte carlo and maximum entropy approach to an imaginary-time formulation of strongly correlated steady-state transport. Phys. Rev. E 2010;82:026701.
[9] Gunnarsson O, Haverkort MW, Sangiovanni G. Analytical continuation of imaginary axis data using maximum entropy. Phys Rev B 2010;81: 155107.
[10] Gunnarsson O, Haverkort MW, Sangiovanni G. Analytical continuation of imaginary axis data for optical conductivity. Phys Rev B 2010;82:165125.
[11] Vidberg HJ, Serene JW. Solving the eliashberg equations by means of n-point pad'e approximants. J Low Temp Phys 1977;29:179–92.
[12] Beach KSD, Gooding RJ, Marsiglio F. Reliable padé analytical continuation method based on a high-accuracy symbolic computation algorithm. Phys Rev B 2000;61:5147–57.
[13] Östlin A, Chioncel L, Vitos L. One-particle spectral function and analytic continuation for many-body implementation in the exact muffin-tin orbitals method. Phys Rev B 2012;86:235107.
[14] Schött J, Locht ILM, Lundin E, Grånäs O, Eriksson O, Di Marco I. Analytic continuation by averaging padé approximants. Phys Rev B 2016; 93:075104.
[15] Creffield CE, Klepfish EG, Pike ER, Sarkar S. Spectral weight function for the half-filled hubbard model: a singular value decomposition approach. Phys Rev Lett 1995;75:517–20.
[16] Tikhonov VSA, Goncharsky A, Yagola A. Numerical methods for the solution of Ill-posed problems. Moscow: Springer-Science+Business Media, B.V; 1995.
[17] Lawson CL, Hanson RJ. Solving least squares problems. Philadelphia: Society for Industrial and Applied Mathematics Philadelphia; 1995.
[18] Krivenko IS, Rubtsov AN. Analytic Continuation of Quantum Monte Carlo Data: Optimal Stochastic Regularization Approach arXiv:cond-mat/0612233.
[19] Mishchenko AS, Prokof'ev NV, Sakamoto A, Svistunov BV. Diagrammatic quantum monte carlo study of the fröhlich polaron. Phys Rev B 2000;62:6317–36.
[20] Vafayi K, Gunnarsson O. Analytical continuation of spectral data from imaginary time axis to real frequency axis using statistical sampling. Phys Rev B 2007;76:035115.
[21] Sandvik AW. Stochastic method for analytic continuation of quantum monte carlo data. Phys Rev B 1998;57:10287–90.
[22] Staar P, Ydens B, Kozhevnikov A, Locquet J-P, Schulthess T. Continuous-pole-expansion method to obtain spectra of electronic lattice models. Phys Rev B 2014;89:245114.
[23] Schött J, van Loon EGCP, Locht ILM, Katsnelson MI, Di Marco I. Phys Rev B 2016; submitted for publication. arXiv:1607.04212.
[24] Gull E, Millis AJ, Lichtenstein AI, Rubtsov AN, Troyer M, Werner P. Continuous-time Monte Carlo methods for quantum impurity models. Rev. Modern Phys 2011;83:349–404.