Parallel algorithm with spectral convergence for nonlinear integro-differential equations

2002 J. Phys. A: Math. Gen. 35 5315

(http://iopscience.iop.org/0305-4470/35/25/311)

INSTITUTE OF PHYSICS PUBLISHING JOURNAL OF PHYSICS A: MATHEMATICAL AND GENERAL

J. Phys. A: Math. Gen. **35** (2002) 5315–5331 PII: S0305-4470(02)33259-1

Parallel algorithm with spectral convergence for nonlinear integro-differential equations

Bogdan Mihaila[1] **and Ruth E Shaw**[2]

[1] Physics Division, Argonne National Laboratory, Argonne, IL 60439, USA
[2] Department of Applied Statistics and Computer Science, University of New Brunswick, Saint John, NB, Canada E2L 4L5

E-mail: bogdan@theory.phy.anl.gov and reshaw@unbsj.ca

Received 25 January 2002
Published 14 June 2002
Online at stacks.iop.org/JPhysA/35/5315

Abstract

We discuss a numerical algorithm for solving nonlinear integro-differential equations, and illustrate our findings for the particular case of Volterra type equations. The algorithm combines a perturbation approach meant to render a linearized version of the problem and a spectral method where unknown functions are expanded in terms of Chebyshev polynomials (El-gendi's method). This approach is shown to be suitable for the calculation of two-point Green functions required in next-to-leading order studies of time-dependent quantum field theory.

PACS numbers: 02.70.−c, 02.30.Mv, 02.60.Jh, 02.70.Bf, 02.60.Nm, 02.60.Lj

1. Introduction

Astrophysical applications related to the physics of the early universe, as well as challenges posed by the physics programs at new heavy ion accelerators, have triggered a renewed interest in the understanding of real time processes in the context of quantum field theory. With the advent of new computer technology and the recent success of new computational schemes, nonequilibrium phenomena which have been previously studied only in the framework mean-field theory [1–3] are now being revisited, and more complex next-to-leading order approaches [4–7] are being used in an attempt to clarify the role played by the rescattering mechanism, which is responsible for driving an out of equilibrium system back to equilibrium. Of particular interest is the study of the dynamics of phase transitions and particle production following a relativistic heavy-ion collision. One way of approaching this study is based on solving Schwinger–Dyson equations within the closed time path (CTP) formulation [8]. This formalism has been recently shown to provide good approximations of the real time evolution of the system both in quantum mechanics and in (1+1)-dimensional classical field theory [9], where direct comparisons with exact calculations can be performed.

0305-4470/02/255315+17$30.00

The key element in carrying out such studies is related to the calculation of the two-point Green function, which is solved for self-consistently with the equations of motion for the fields. The two-point Green function gives rise to Volterra type integral or integro-differential equations. In the process of extending our study to encompass a higher number of spatial dimensions, i.e. 2+1 and 3+1 field theory, we are faced with the challenge of coping with constraints dictated both by storage and time-related computational limits. Thus our interest lies in designing algorithms which feature spectral convergence in order to achieve convergence with minimum storage requirements. In addition, we also desire these algorithms to scale when ported to massively multiprocessor (MPP) machines, so that solutions can be obtained in a reasonable amount of time.

Algorithms for Volterra integral and integro-differential equations usually start out at the lower end of the domain, a, and march out from $x = a$, building up the solution as they go [10]. Such methods are *serial* by nature, and are, in general, not suitable for parallel implementation on a MPP machine. Even so, clever approaches to already existing methods can provide algorithms that take advantage of a parallel processing computer: Shaw [11] has shown recently that once the starting values of the approximation are obtained, one can design a *global* approach where successive approximations of the solution over the entire domain $x \in [a, b]$ can be evaluated simultaneously.

In a recent paper [12] one of us has discussed a spectral method [13] of solving some types of equations of interest for the study of time-dependent nonequilibrium problems in quantum field theory. The gist of the method consists in expanding out the unknown function in terms of Chebyshev polynomials on a suitable grid, thus reducing the problem to finding the numerical solution of a system of linear equations. The main advantage of this method over standard finite-difference type methods resides in the spectral character of its convergence. This is related in part to the fact that Chebyshev type methods use a non-uniform grid, while finite-difference methods require a uniform grid. Usually, there is a trade-off between computational time and storage requirements, and a balanced solution must be reached on a case-by-case basis. Spectral methods are more expensive per point as the matrices may be considerably denser than in the finite-difference case, but we require considerably fewer grid points in order to achieve the same degree of accuracy. By expanding the unknown function on a compact support in Chebyshev polynomials and using a partition of the domain based either on the set of $(N + 1)$ extrema or on the set of N zeros of $T_N(x)$—the Chebyshev polynomial of first kind of degree N—we in fact replace a continuous problem by a discrete one. For non-singular functions the discrete orthogonality and completeness relations for Chebyshev polynomials at the above grid points assure a *de facto* exact expansion for an *arbitrary* finite value N. In practice, however, one has to compute derivatives and integrals of the unknown function at the collocation points, and the Chebyshev expansion provides only an approximation for these subsequent computations. These errors, together with the finite accuracy of numerical methods needed in conjunction with the Chebyshev expansion, conspire in order to deteriorate the accuracy of the solution at very small values of N.

The paper is organized as follows. In section 2, for comparison purposes, we start by reviewing a finite-difference approach for the numerical solution of Volterra type integro-differential equations. We review the general framework of the Chebyshev-expansion method in section 3, and illustrate our approach for the case of Volterra integro-differential equations. In section 4, we present a complete assessment of the convergence and computational cost of the proposed method for the case of a test problem, and compare with results obtained via the finite-difference method. In section 5, we discuss the relevant aspects of a large-scale calculation arising in the study of time-dependent quantum field theory, for which our numerical strategy is particularly suitable. We present our conclusions in section 6.

2. Stable multi-step method for Volterra type equations

The type of problems arising in the study of time-dependent nonequilibrium quantum field theory via a Schwinger–Dyson equation approach, can be formally reduced to the general case of a nonlinear Volterra integro-differential equation. Direct methods for solving nonlinear Volterra integral and integro-differential equations are inherently serial and therefore have not received much attention for use on a parallel computer. It is worth mentioning here the work of Crisci *et al* [14] who concentrated on the stability aspects of parallel iteration of Volterra-Runge–Kutta (VRK) methods for solving Volterra integral equations on parallel computers. VRK methods are step-by-step methods and can take advantage of parallel architecture. Sommeijer *et al* [15] covered the stability of parallel block methods for ordinary differential equations (ODE) and included equations of the integro-differential type in their discussion.

We summarize here a recent parallel algorithm [11] which concentrates on modifying the algorithmic side of the numerical solution process for use on a parallel processor while consciously utilizing methods that are known to be stable. The algorithm is in effect an example of a higher-order finite-difference approach, and we use this approach to compare with the spectral method presented later in this paper.

For illustration, let us consider a first-order nonlinear Volterra integro-differential equation of the form

$$\boldsymbol{y}'(x) = F[x, \boldsymbol{y}, Z[x; \boldsymbol{y}]] \qquad x \in [a, b] \tag{1}$$

with

$$Z[x; \boldsymbol{y}] = \int_a^x K[x, t; \boldsymbol{y}(t)]\,\mathrm{d}t \tag{2}$$

and subject to the initial condition

$$\boldsymbol{y}(a) = y_0. \tag{3}$$

Let I_N be a partition of $I = [a, b]$, where $I_N = \{x_N = a + nh, n = 0(1)N, Nh = (b - a)\}$. The problem is to find approximations y_n to the solution $\boldsymbol{y}(x_n)$ of equations (1)–(3) for each $x_n \in I_N$. A k-step method for an integro-differential equation of the form (1) is given by

$$y_{n+1} = y_n + h \sum_{j=0}^{k} w_j F(x_{n-j}, y_{n-j}, z_{n-j}) \qquad n = k(1)N \tag{4}$$

where

$$z_{n-j} = h \sum_{i=0}^{n-j} c_{n-j,i} K(x_{n-j}, x_i, y_i) \qquad j = 0(1)k \quad y_0 = \boldsymbol{y}(a). \tag{5}$$

The weights w_i depend on the k-step method selected and the weights $c_{i,j}$ are those of a standard quadrature formula for integrating a function whose value is known at equally spaced steps, such as a Newton–Cotes or Newton–Gregory quadrature rule. For our multi-step ($k = 4$) method [10] we choose the fourth-order Adams–Bashforth predictor

$$\begin{aligned} y_{k+1}^0 = y_k + \frac{h}{24}[&55F(x_k, y_k, z_k) - 59F(x_{k-1}, y_{k-1}, z_{k-1}) + 37F(x_{k-2}, y_{k-2}, z_{k-2}) \\ &- 9F(x_{k-3}, y_{k-3}, z_{k-3})] \end{aligned} \tag{6}$$

and the Adams–Moulton corrector

$$\begin{aligned} y_{k+1} = y_k + \frac{h}{24}\big[&9F\big(x_{k+1}, y_{k+1}^0, z_{k+1}\big) + 19F(x_k, y_k, z_k) - 5F(x_{k-1}, y_{k-1}, z_{k-1}) \\ &+ F(x_{k-2}, y_{k-2}, z_{k-2})\big] \end{aligned} \tag{7}$$

while the integral term (2) is calculated based on the Newton–Gregory quadrature formula. We use a fourth-order Runge–Kutta method in order to start out the calculation.

In order to make the algorithm suitable for parallel processing, it is useful to recall that a standard quadrature method based on an uniform grid for the integral term z_i requires knowledge of the integrand function at the abscissas in the interval $[x_0, x_i]$. This is obviously a serial process and not a good candidate for parallelization. It can be observed, however, that once the starting values are obtained, *all* approximations z_i with $i = 0(1)k-1$ can simultaneously be evaluated up to and including x_{k-1}. After that, once a value of y_j corresponding to a new step x_j is established via the predictor–corrector method, all values z_i with $i = j(1)N$ can also be evaluated simultaneously. This observation makes the following algorithm possible:

(i) find the starting values (y_i, z_i) with $i = 0(1)k-1$;
(ii) *do* $i = k, N$:
  add contributions to z_i corresponding to (x_j, y_j), where $j = 0(1)k-1$;
(iii) *do* $i = k, N$:
  (a) predict y_i;
  (b) estimate z_i from (x_i, y_i);
  (c) correct y_i;
  (d) *do* $j = i, N$:
    update z_j by adding the contribution corresponding to (x_i, y_i).

The above numerical algorithm is implemented using the OpenMP style directives for the Portland Group's pgf77 FORTRAN compiler, and reportedly shows good scalability on a shared-memory multiprocessor. The speedup of the finite-difference method is best for a large number of grid points which, correspondingly, gives a better solution approximation. For example, with $N = 5120$ and 4 processors the speedup is 3.86, a good measure of processor utilization.

While the preceding algorithm performs well on a shared memory platform, it does not port easily to an MPP machine. Before we comment on the efficiency of the algorithm, let us make two general comments: firstly, we denote by T_{calc} and T_{comm} the time required to perform a floating-point operation and the time required to send a floating-point number, respectively. Secondly, we will ignore for simplicity the effect of message sizes on communication costs, and assume throughout that the ratio $T_{\text{comm}}/T_{\text{calc}}$ is independent of N.

Returning now, to our proposed algorithm, we remark that the communication cost for the corresponding implementation involves only the integral terms. Even so, using the message-passing interface (MPI) protocol the communication cost is $4 \log N$ for the starting values and up to N^2 for the remainder of the algorithm which gives a total of $(N^2 + 4 \log N)T_{\text{comm}}$. The total number of flops depends on the specific application but a reasonable measure is the number of function evaluations which is given by $(N^2 + 4N)T_{\text{calc}}$. The ratio of communication to computation

$$\frac{N^2 + 4 \log N}{N^2 + 4N} \frac{T_{\text{comm}}}{T_{\text{calc}}}$$

approaches a *constant* value as N gets larger. The communication overhead problem can be relaxed by employing a spectral method discussed in the following section, the improvement being especially significant for a multi-dimensional problem of the type required by our nonequilibrium quantum field theory calculations [9].

3. Spectral method with Chebyshev polynomials

Consider the $N+1$ extrema of the Chebyshev polynomial of the first kind of degree N, $T_N(x)$. This set defines a non-uniform grid in the interval $[-1, 1]$, as

$$\tilde{x}_k = \cos\left(\frac{\pi k}{n}\right) \qquad k = 0(1)N. \tag{8}$$

On this grid, the Chebyshev polynomials of degree $i < n$ obey discrete orthogonality relations

$$\sum_{k=0}^{N}{}'' T_i(\tilde{x}_k) T_j(\tilde{x}_k) = \beta_i \delta_{ij} \tag{9}$$

where the constants β_i are

$$\beta_i = \begin{cases} \dfrac{N}{2} & i \neq 0, N \\ N & i = 0, N. \end{cases}$$

Here, the summation symbol with double primes denotes a sum with both the first and last terms halved. We approximate an arbitrary continuous function of bounded variation $f(x)$ in the interval $[-1, 1]$, as

$$f(x) \approx \sum_{j=0}^{N}{}'' b_j T_j(x) \tag{10}$$

with

$$b_j = \frac{2}{N} \sum_{k=0}^{N}{}'' f(\tilde{x}_k) T_j(\tilde{x}_k) \qquad j = 0(1)N. \tag{11}$$

Equation (10) is exact at x equal to $\tilde{x}_k$ given by equation (8). Based on equation (10), we can also approximate derivatives and integrals as

$$f'(x) \approx \sum_{k=0}^{N}{}'' f(\tilde{x}_k) \frac{2}{N} \sum_{j=0}^{N}{}'' T_j(\tilde{x}_k) T_j'(x). \tag{12}$$

and

$$\int_{-1}^{x} f(t)\,\mathrm{d}t \approx \sum_{k=0}^{N}{}'' f(\tilde{x}_k) \frac{2}{N} \sum_{j=0}^{N}{}'' T_j(\tilde{x}_k) \int_{-1}^{x} T_j(t)\,\mathrm{d}t. \tag{13}$$

In matrix format, we have

$$\left[\int_{-1}^{x} f(t)\,\mathrm{d}t\right] \approx \tilde{S}[f] \tag{14}$$

$$[f'(x)] \approx \tilde{D}[f]. \tag{15}$$

The elements of the column matrix $[f]$ are given by $f(\tilde{x}_k)$, $k = 0(1)N$. The right-hand side of equations (14) and (15) gives the values of the integral $\int_{-1}^{x} f(t)\,\mathrm{d}t$ and the derivative $f'(x)$ at the corresponding grid points, respectively. The actual values of the elements of the matrices $\tilde{S}$ and $\tilde{D}$ can be derived using equations (12) and (13).

In order to illustrate the Chebyshev algorithm, we consider again the case of a first-order nonlinear Volterra integro-differential equation of the form

$$\boldsymbol{y}'(x) = F[x, \boldsymbol{y}, Z[x; \boldsymbol{y}]] \qquad x \in [a, b]$$

$$Z[x; \boldsymbol{y}] = \int_{a}^{x} K[x, t; \boldsymbol{y}(t)]\,\mathrm{d}t$$

with the initial condition

$$\boldsymbol{y}(a) = y_0.$$

Here we make no explicit restrictions on the actual form of the function $F[x, \boldsymbol{y}, Z[x; \boldsymbol{y}]]$, so both linear and nonlinear equations are included. We determine the unknown function $\boldsymbol{y}(x)$ using a perturbation approach: we start with an initial guess of the solution $\boldsymbol{y}_0(x)$ that satisfies the initial condition $\boldsymbol{y}_0(a) = y_0$, and write

$$\boldsymbol{y}(x) = \boldsymbol{y}_0(x) + \epsilon(x)$$

with $\epsilon(x)$ being a variation obeying the initial condition

$$\epsilon(a) = 0. \tag{16}$$

Hence, the original problem reduces to finding the perturbation $\epsilon(x)$, and improving the initial guess in an iterative fashion.

We use the Taylor expansion of $F[x, \boldsymbol{y}, Z[x; \boldsymbol{y}]]$ about $\boldsymbol{y}(x) = \boldsymbol{y}_0(x)$ and keep only the linear terms in $\epsilon(x)$ to obtain an equation for the variation $\epsilon(x)$,

$$\begin{aligned}\epsilon'(x) - \left.\frac{\partial F[x, \boldsymbol{y}, Z[x; \boldsymbol{y}]]}{\partial \boldsymbol{y}(x)}\right|_{\boldsymbol{y}(x)=\boldsymbol{y}_0(x)} \epsilon(x) \\ - \left.\frac{\partial F[x, \boldsymbol{y}, Z[x; \boldsymbol{y}]]}{\partial Z[x; \boldsymbol{y}]}\right|_{\boldsymbol{y}(x)=\boldsymbol{y}_0(x)} \int_a^x \left.\frac{\partial K[x, t; \boldsymbol{y}(t)]}{\partial \boldsymbol{y}(x)}\right|_{\boldsymbol{y}(x)=\boldsymbol{y}_0(x)} \epsilon(t)\,\mathrm{d}t \\ = -\boldsymbol{y}_0'(x) + F[x, \boldsymbol{y}_0(x), Z[x; \boldsymbol{y}_0(x)]].\end{aligned} \tag{17}$$

Equation (17) is of the general form (18)

$$\epsilon'(x) = q[x, \epsilon(x)] + r(x) \tag{18}$$

where

$$\begin{aligned}q[x, \epsilon(x)] = \left.\frac{\partial F[x, \boldsymbol{y}, Z[x; \boldsymbol{y}]]}{\partial \boldsymbol{y}(x)}\right|_{\boldsymbol{y}(x)=\boldsymbol{y}_0(x)} \epsilon(x) \\ + \left.\frac{\partial F[x, \boldsymbol{y}, Z[x; \boldsymbol{y}]]}{\partial Z[x; \boldsymbol{y}]}\right|_{\boldsymbol{y}(x)=\boldsymbol{y}_0(x)} \int_a^x \left.\frac{\partial K[x, t; \boldsymbol{y}(t)]}{\partial \boldsymbol{y}(x)}\right|_{\boldsymbol{y}(x)=\boldsymbol{y}_0(x)} \epsilon(t)\,\mathrm{d}t\end{aligned}$$

and

$$r(x) = -\boldsymbol{y}_0'(x) - F[x, \boldsymbol{y}_0(x), Z[x; \boldsymbol{y}]]$$

together with the initial condition given by (16). We replace equations (18) and (16) by an integral equation, obtained by integrating equation (18) and using the initial condition (16) to choose the lower bound of the integral. We obtain

$$\epsilon(x) = \int_a^x q[t, \epsilon(t)]\,\mathrm{d}t + \int_a^x r(t)\,\mathrm{d}t \tag{19}$$

which is in fact a *linear* Volterra integral equation of the second kind. Using the techniques developed in the previous section to calculate integrals, the integral equation (19) can be transformed into a linear system of equations. A practical implementation of this algorithm is illustrated via a test problem in the following section.

4. Test problem

Following Shaw [11], we consider the test problem

$$\boldsymbol{y}(x) = x\,\mathrm{e}^{1-\boldsymbol{y}(x)} - \frac{1}{(1+x)^2} - x - \int_0^x \frac{x}{(1+t)^2}\,\mathrm{e}^{1-\boldsymbol{y}(t)}\,\mathrm{d}t \tag{20}$$

$$\boldsymbol{y}(0) = y_0 = 1 \qquad x \in [0,1] \tag{21}$$

which has the exact solution

$$\boldsymbol{y}(x) = \frac{1}{1+x}. \tag{22}$$

We shall use the initial guess $\boldsymbol{y}_0(x) = y_0\cos(x)$, so that $\boldsymbol{y}_0(0) = y_0$. The equation for the variation $\epsilon(x)$ is

$$\begin{aligned}\epsilon(x) - \int_0^x t\,\mathrm{e}^{1-y_0(t)}\epsilon(t)\,\mathrm{d}t + \int_0^x \mathrm{d}s \int_0^s \frac{s\,\mathrm{e}^{1-y_0(t)}}{(1+t)^2}\epsilon(t)\,\mathrm{d}t \\ = -\boldsymbol{y}_0(x) + y_0 + \int_0^x \left[t\,\mathrm{e}^{1-y_0(t)} - \frac{1}{(1+t)^2} - t\right]\mathrm{d}t - \int_0^x \mathrm{d}s \int_0^s \frac{s\,\mathrm{e}^{1-y_0(t)}}{(1+t)^2}\,\mathrm{d}t.\end{aligned} \tag{23}$$

In matrix format and using the Chebyshev expansion presented above, the variation $\epsilon(x)$ will be obtained as the solution of linear system of equations

$$A[\epsilon] = C \tag{24}$$

with matrices A and C given as

$$A_{ij} = \delta_{ij} - \tilde{S}_{ij}\left[t\,\mathrm{e}^{1-y_0(t)}\right]_j + \tilde{S}_{ik}\tilde{x}_k\tilde{S}_{kj}\left[\frac{\mathrm{e}^{1-y_0(t)}}{(1+t)^2}\right]_j \qquad i,\,j = 0(1)N$$

$$C_i = -[\boldsymbol{y}_0(t)]_i + y_0 + \tilde{S}_{ik}\left[t\,\mathrm{e}^{1-y_0(t)} - \frac{1}{(1+t)^2} - t\right]_k - \tilde{S}_{ik}\tilde{x}_k\tilde{S}_{k\ell}\left[\frac{\mathrm{e}^{1-y_0(t)}}{(1+t)^2}\right]_\ell.$$

From a computational point of view, the computer time is spent initializing the matrix elements A_{ij} and C_j on one hand, and finding the solution of (24) on the other. On the first matter, the calculation decouples nicely, and once we have the vector $[y_0]$, we can calculate $\{C_i, A_{ij},\, j = 0(1)N\}$ in parallel for $i = 0(1)N$. The algorithm is as follows:

(i) calculate $[y_0] = [y_0] + [\epsilon]$;
(ii) broadcast $[y_0]$;
(iii) *do* $i = 0, N$:
  (a) master to slave: send i;
  (b) slave: compute $\{C_i, A_{ij},\, j = 0(1)N\}$;
  (c) slave to master: return $\{C_i, A_{ij},\, j = 0(1)N\}$.

Regarding the second step, i.e. solving the linear system of equations, the best choice is to use the machine specific subroutines, which generally outperform hand-coded solutions. When such subroutines are not available, as in the case of a Linux based PC cluster for instance, one can use one of the MPI implementations available on the market. We shall see that the efficiency of the equation solver is critical to the success of the parallel implementation of the Chebyshev-expansion approach. In order to illustrate this aspect, we perform two calculations, first using a LU factorization algorithm, and second using an iterative biconjugate gradient algorithm. These are standard algorithms [10] for solving systems of linear equations, but their impact on the general efficiency of the approach is quite different.

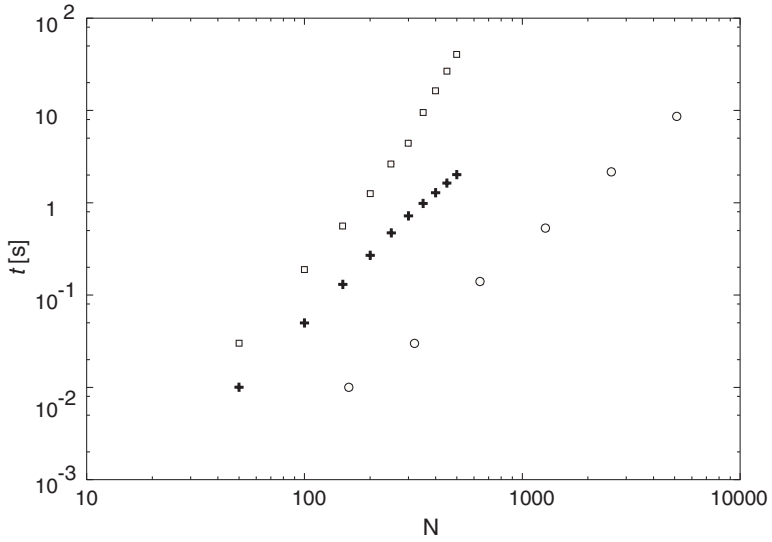

Figure 1. Average CPU time versus the number of grid points for the Chebyshev-expansion approach using either the LU decomposition (squares) or the biconjugate gradient method (crosses) and finite-difference approach (circles).

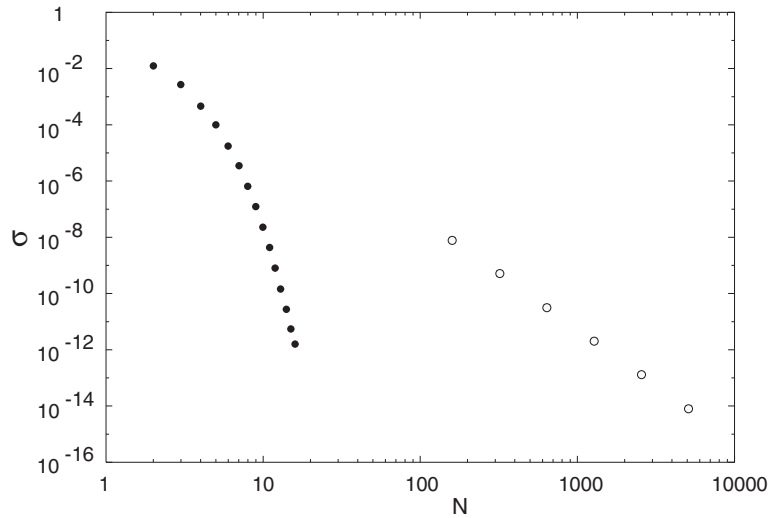

Figure 2. Convergence of the Chebyshev result (full circles) compared with the finite-difference result (open circles), versus the number of grid points.

4.1. Serial case

Figure 1 depicts the average CPU time required to complete the calculation for the various methods. Figure 2 illustrates the convergence of the two numerical methods. The spectral character of the method based on Chebyshev polynomials allows for an excellent representation of the solution for $N > 12$. We base our findings on a $\sigma < 10^{-10}$ criterion, where σ denotes

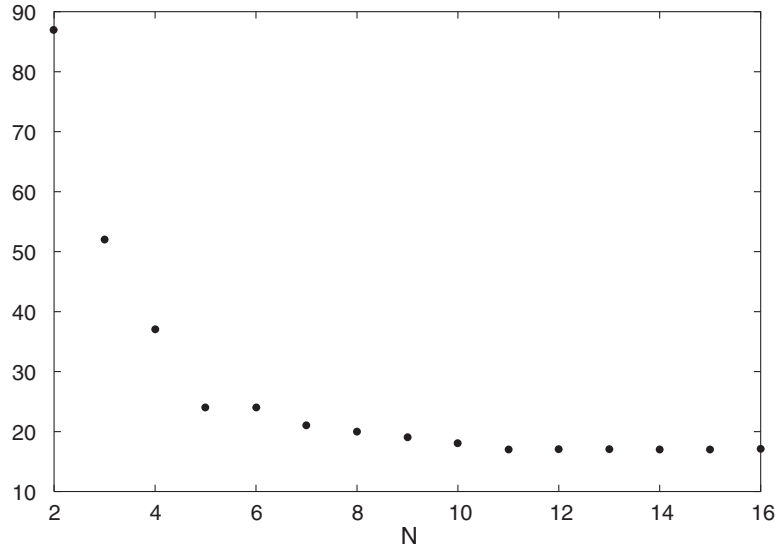

Figure 3. Number of iterations versus the number of grid points for the Chebyshev method.

the sum of all absolute departures of the calculated values from the exact ones, at the grid points.

The number of iterations required to achieve the desired accuracy in the Chebyshev case is depicted in figure 3. The number of iterations becomes flat for $N > 12$, and stays constant (17 iterations) even for very large values of N. The higher number of iterations corresponding to the lower values of N represents an indication of an insufficient number of Chebyshev grid points: the exact solution cannot be accurately represented as a polynomial of degree N for $x \in [0, 1]$. It is interesting to note that for $N = 12$–16, a reasonable lower domain for the representation of the solution using Chebyshev polynomials, the reported CPU time is so small that for our test problem there is no real justification for porting the algorithm to a MPP machine. This situation will change for multi-dimensional problems such as those encountered in our nonequilibrium quantum field theory studies.

4.2. Parallel case

The LU factorization algorithm is an algorithm of order N^3 and consequently, most of the CPU time is spent solving the linear system of equations (see figure 4). As a consequence, a parallel implementation of the LU algorithm is very difficult. Figure 5 shows how the average CPU time changes with the available number of processors. Here we use a very simple MPI implementation of the LU algorithm as presented in [16]. Even though we could certainly achieve better performance by employing a sophisticated LU equation solver, the results are typical. Since the actual size of the matrices involved is small, the communication overhead is overwhelming and the execution time does not scale with the number of processors.

Fortunately, even for dense matrices and small values of the number of grid points N, one can achieve a good parallel efficiency. By employing an iterative method such as the iterative biconjugate gradient method, one can render the time required to solve the system of linear equations negligible compared with the time required to initialize the relevant matrices, which in turn is only slightly more expensive than the initialization process of the LU factorization

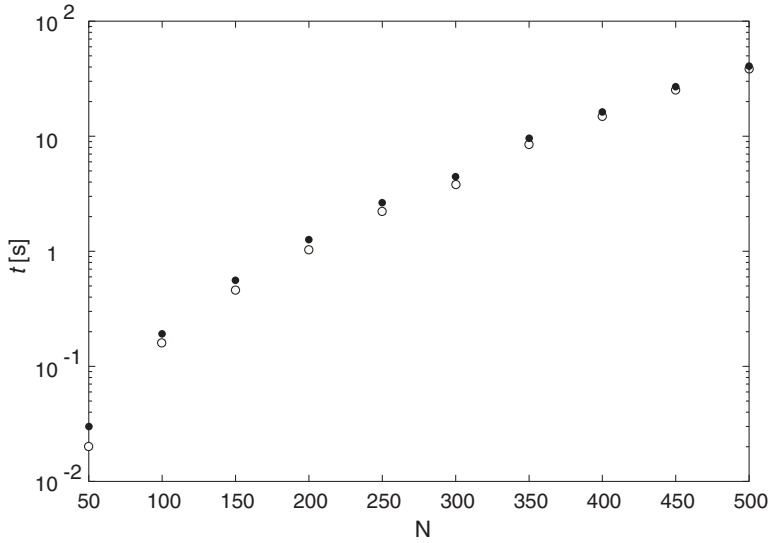

Figure 4. Total CPU time (full circles) and CPU time spent carrying out the LU decomposition (open circles), versus the number of grid points (1 CPU case).

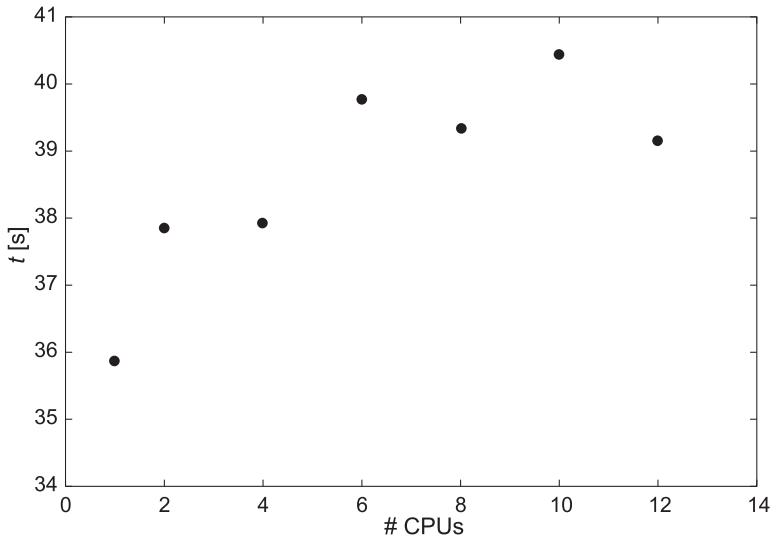

Figure 5. Scaling of the average CPU time with the number of available processors for the Chebyshev-expansion approach and the LU factorization algorithm ($N = 500$).

algorithm. The initialization process can be parallelized using the algorithm presented above and the results are depicted in figure 6.

It appears that by using the biconjugate gradient method the efficiency of the parallel code has improved considerably. However, the average CPU time saturates to give an overall speedup of 3.5. This can be understood by analysing the computation and communication requirements for our particular problem. The calculation cost to initialize the matrices A and C is roughly given by the number of floating-point multiplications and additions

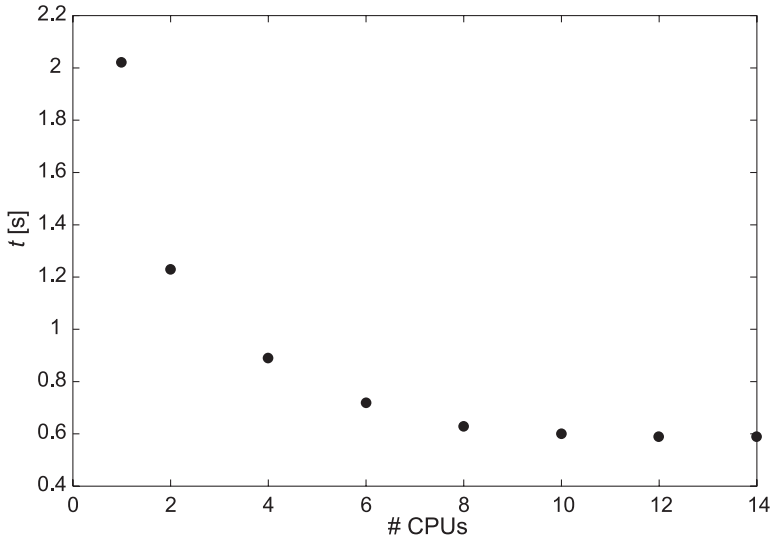

Figure 6. Scaling of the average CPU time with the number of available processors for the Chebyshev-expansion approach and the biconjugate gradient algorithm ($N = 500$).

$(7N^2 + 3N)T_{\text{calc}}$, while the communication cost is given by $(N^2 + 2N)T_{\text{comm}}$. Therefore, the ratio of communication to computation is

$$\frac{N^2 + 2N}{7N^2 + 3N} \frac{T_{\text{comm}}}{T_{\text{calc}}}.$$

As in the finite-difference case, this ratio approaches a *constant* value as N gets larger and it becomes apparent that the communication overhead is still a problem.

However, multi-dimensional applications such as those presented in [9] require complicated matrix element calculation. In such cases, the process of initializing the matrices A and C is quite involved, and the ratio of the communication time relative to the computation time becomes favourable. In addition, the matrix A becomes sparse and the size of the linear system of equations is substantially larger, thus one can also take advantage of existing parallel implementation of the iterative biconjugate gradient algorithm [17]. Such problems benefit heavily from an adequate parallelization of the code. We will discuss such an example in the following section.

5. Volterra-like integral equations for a two-point Green function

Schwinger, Bakshi, Mahanthappa and Keldysh [8] have established how to formulate an initial value problem in quantum field theory. The formalism is based on a generating functional, and the evolution of the density matrix requires both a forward evolution from zero to t and a backward one from t to zero. This involves [18] both positive and negative time ordered operators in the evolution of the observable operators and the introduction of two currents into the path integral for the generating functional. Time integrals are then replaced by integrals along the closed time path (CTP) in the complex time plane shown in figure 7. We have

$$\int_{\mathcal{C}} F(t)\,\mathrm{d}t = \int_{0:\mathcal{C}_+}^{\infty} F_+(t)\,\mathrm{d}t - \int_{0:\mathcal{C}_-}^{\infty} F_-(t)\,\mathrm{d}t. \tag{25}$$

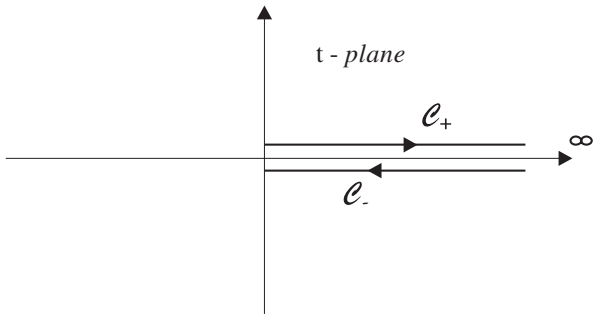

Figure 7. Complex time contour $\mathcal{C}$ for the closed time path integrals.

Using the CTP contour, the full closed time path Green function for the two-point functions is

$$\mathcal{G}(t,t') = \mathcal{G}_{>}(t,t')\Theta_{\mathcal{C}}(t,t') + \mathcal{G}_{<}(t,t')\Theta_{\mathcal{C}}(t',t)$$

in terms of the Wightman functions, $\mathcal{G}_{>,<}(t,t')$, where the CTP step function $\Theta_{\mathcal{C}}(t,t')$ is defined by

$$\Theta_{\mathcal{C}}(t,t') = \begin{cases} \Theta(t,t') & \text{for } t \text{ on } \mathcal{C}_+ \text{ and } t' \text{ on } \mathcal{C}_+ \\ 0 & \text{for } t \text{ on } \mathcal{C}_+ \text{ and } t' \text{ on } \mathcal{C}_- \\ 1 & \text{for } t \text{ on } \mathcal{C}_- \text{ and } t' \text{ on } \mathcal{C}_+ \\ \Theta(t',t) & \text{for } t \text{ on } \mathcal{C}_- \text{ and } t' \text{ on } \mathcal{C}_-. \end{cases} \tag{26}$$

For complete details of this formalism and various applications, we refer the reader to the original literature [8, 18], and we confine ourselves to discussing how our Chebyshev-expansion approach is applied to the computation of the two-point Green function.

For simplicity, we consider now the quantum mechanical limit of quantum field theory (0+1 dimensions). In this limit, we are generally faced with the problem of numerically finding the solution of equation

$$\mathcal{G}(t,t') = G(t,t') - \int_{\mathcal{C}} \mathrm{d}t'' Q(t,t'')\mathcal{G}(t'',t'). \tag{27}$$

Here, the Green functions, $\mathcal{G}(t,t')$ and $G(t,t')$, are symmetric in the sense that $\mathcal{G}_{>}(t,t') = \mathcal{G}_{<}(t',t)$, and obey the additional condition

$$\mathcal{G}_{>,<}(t,t') = -\mathcal{G}^*_{<,>}(t,t') = \mathcal{G}_{<,>}(t',t). \tag{28}$$

The function $Q(t,t')$ obeys less stringent symmetries

$$Q_{>,<}(t,t') = -Q^*_{<,>}(t,t') \neq Q_{<,>}(t',t) \tag{29}$$

which is always the case when $Q(t,t')$ has the form

$$Q(t,t') = \int_{\mathcal{C}} \mathrm{d}t'' A(t,t'')B(t'',t') \tag{30}$$

where $A(t,t')$ and $B(t,t')$ satisfy (28).

We can further write equation (28) as

$$\mathrm{Re}\{\mathcal{G}_{>}(t,t')\} = -\mathrm{Re}\{\mathcal{G}_{<}(t,t')\} \tag{31}$$

$$\mathrm{Im}\{\mathcal{G}_{>}(t,t')\} = \mathrm{Im}\{\mathcal{G}_{<}(t,t')\} \tag{32}$$

or

$$\mathcal{G}_>(t,t') - \mathcal{G}^*_<(t,t') = 2\,\mathrm{Re}\{\mathcal{G}_>(t,t')\} \tag{33}$$

$$\mathcal{G}_>(t,t') + \mathcal{G}^*_<(t,t') = 2\,\mathrm{Im}\{\mathcal{G}_>(t,t')\}. \tag{34}$$

Hence, a Green function $\mathcal{G}(t,t')$ is fully determined by the component $\mathcal{G}_>(t,t') = \mathrm{Re}\{\mathcal{G}_>(t,t')\} + \mathrm{i}\,\mathrm{Im}\{\mathcal{G}_>(t,t')\}$, with $t' \leqslant t$. Thus, in order to obtain the solution of equation (27), we only need to solve

$$\mathcal{G}_>(t,t') = G_>(t,t') - 2\int_0^t \mathrm{d}t''\,\mathrm{Re}\{Q_>(t,t'')\}\mathcal{G}_>(t'',t') + 2\int_0^{t'} \mathrm{d}t''\,Q_>(t,t'')\,\mathrm{Re}\{\mathcal{G}_>(t'',t')\}. \tag{35}$$

We separate the real and the imaginary parts of (35) and obtain the system of integral equations

$$\begin{aligned}\mathrm{Re}\{\mathcal{G}_>(t,t')\} = \mathrm{Re}\{G_>(t,t')\} &- 2\int_0^t \mathrm{d}t''\,\mathrm{Re}\{Q_>(t,t'')\}\,\mathrm{Re}\{\mathcal{G}_>(t'',t')\} \\ &+ 2\int_0^{t'} \mathrm{d}t''\,\mathrm{Re}\{Q_>(t,t'')\}\,\mathrm{Re}\{\mathcal{G}_>(t'',t')\}\end{aligned} \tag{36}$$

$$\begin{aligned}\mathrm{Im}\{\mathcal{G}_>(t,t')\} = \mathrm{Im}\{G_>(t,t')\} &- 2\int_0^t \mathrm{d}t''\,\mathrm{Re}\{Q_>(t,t'')\}\,\mathrm{Im}\{\mathcal{G}_>(t'',t')\} \\ &+ 2\int_0^{t'} \mathrm{d}t''\,\mathrm{Im}\{Q_>(t,t'')\}\,\mathrm{Re}\{\mathcal{G}_>(t'',t')\}.\end{aligned} \tag{37}$$

The above system of equations must be solved for $t' \leqslant t$. The two equations are independent, which allows us to solve first for the real part of $\mathcal{G}_>(t,t')$ and then use this result to derive the imaginary part of $\mathcal{G}_>(t,t')$.

Despite their somewhat unusual form, the above equations are two-dimensional Volterra-like integral equations and our general discussion regarding the Chebyshev spectral method applies. We will perform a multi-step implementation of the formalism. Let

$$t_i = t_{i_0(N-1)+i_1} \qquad 1 \leqslant i_1 \leqslant N$$

be the grid location corresponding to the collocation point i_1 of the interval labelled $i_0 + 1$. Then, the discrete correspondent of equation (35) is

$$\begin{aligned}\mathcal{G}_>(t_i,t_j) = G_>(t_i,t_j) &- \sum_{k_0=0}^{i_0-1}\sum_{k_1=1}^{N}\left[2\tilde{S}_{Nk_1}\right]\mathrm{Re}\left\{Q_>\left(t_i, t_{k[=k_0(N-1)+k_1]}\right)\right\}\mathcal{G}_>(t_k,t_j) \\ &- \sum_{k_1=1}^{N}\left[2\tilde{S}_{i_1k_1}\right]\mathrm{Re}\left\{Q_>\left(t_i, t_{k[=i_0(N-1)+k_1]}\right)\right\}\mathcal{G}_>(t_k,t_j) \\ &+ \sum_{k_0=0}^{j_0-1}\sum_{k_1=1}^{N}\left[2\tilde{S}_{Nk_1}\right]Q_>\left(t_i, t_{k[=k_0(N-1)+k_1]}\right)\mathrm{Re}\{\mathcal{G}_>(t_k,t_j)\} \\ &+ \sum_{k_1=1}^{N}\left[2\tilde{S}_{j_1k_1}\right]Q_>\left(t_i, t_{k[=j_0(N-1)+k_1]}\right)\mathrm{Re}\{\mathcal{G}_>(t_k,t_j)\}\end{aligned} \tag{38}$$

with $t_j \leqslant t_i$.

We will refer now to figures 8 and 9. Equation (39) involves values of $\mathcal{G}_>(t_k,t_j)$, for which $t_j > t_k$. In such cases, we use the symmetry $\mathcal{G}^*_>(t_j,t_k)$, which relates to the values of

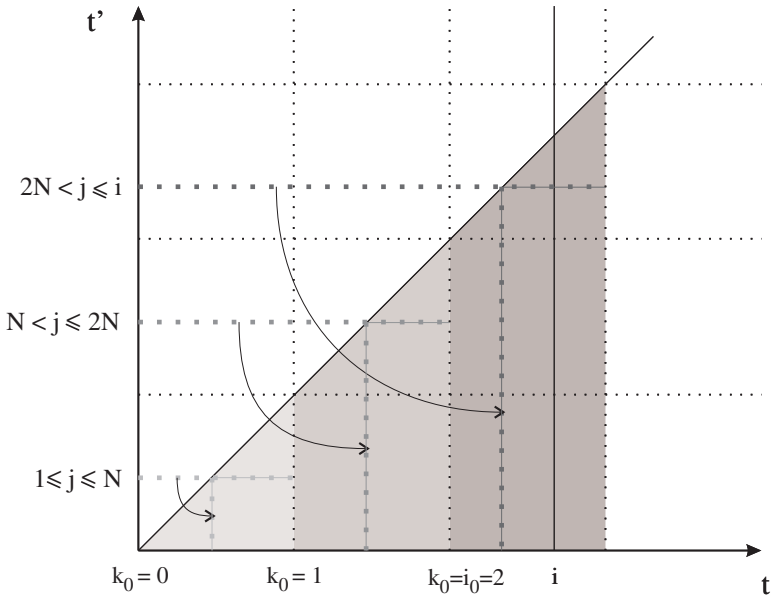

Figure 8. $\mathcal{G}(t_k, t_j)$ contributions to the integral $\int_0^{t_i} Q(t_i, t_k)\mathcal{G}(t_k, t_j)\,\mathrm{d}t_k$ with $t_j \leqslant t_i$.

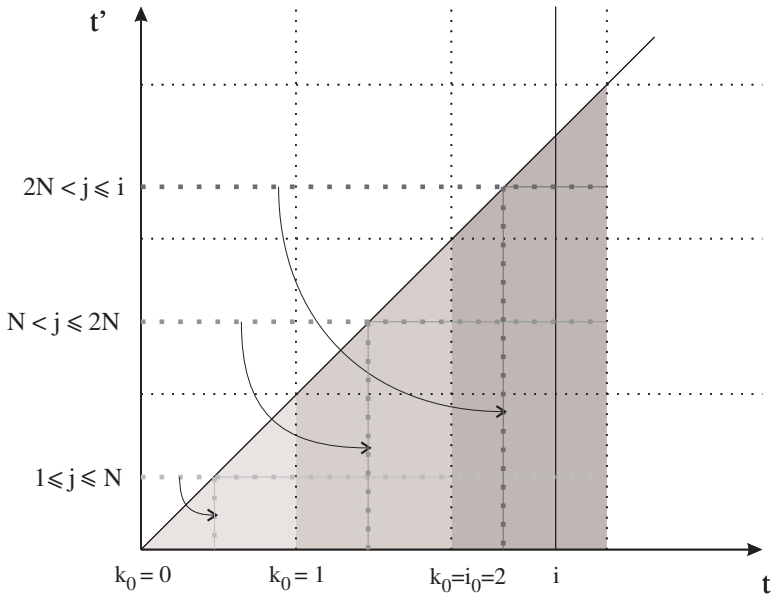

Figure 9. $\mathcal{G}(t_k, t_j)$ contributions to the integral $\int_0^{t_j} Q(t_i, t_k)\mathcal{G}(t_k, t_j)\,\mathrm{d}t_k$ with $t_j \leqslant t_i$.

the two-point function located in the domain of interest. For the time interval $(i_0 + 1)$ the size of the linear system of equations we need to solve is

$$\tfrac{1}{2}(i_0+1)(N-1)[(i_0+1)(N-1)+1] - \tfrac{1}{2}i_0(N-1)[i_0(N-1)+1]$$
$$= i_0(N-1)^2 + \tfrac{1}{2}N(N-1)$$

or of order $\left(i_0 + \frac{1}{2}\right)(N-1)^2$. In practice, the value of N is taken between 16 and 32.

Table 1. Summary regarding the calculation of $\mathrm{Re}\,\mathcal{G}(t_i, t_j)$ at step $i_0 + 1$.

Integral	Domain	Non-zero elements	Additions	Multiplications
$\int_0^{t_i} \mathrm{d}t_k$	$j \leqslant i_0(N-1)$	N	$i_0 N$	$(2i_0+1)N$
$\int_0^{t_j} \mathrm{d}t_k$	$j \leqslant i_0(N-1)$	0	$(j_0+1)N$	$(2j_0+1)N$
Total		$N+1$	$(i_0+j_0+1)N+1$	$2(i_0+j_0+1)N$
$\int_0^{t_i} \mathrm{d}t_k$	$j > i_0(N-1)$	$(i_0+1)(N-1)+1$	i_0	$(i_0+1)N$
$\int_0^{t_j} \mathrm{d}t_k$	$j > i_0(N-1)$	$(i_0+1)(N-1)+1$	i_0	$(i_0+1)N$
Total		$(i_0+1)(N-1)+2$	$(i_0+1)(N+1)$	$2(i_0+1)N$

Table 2. Summary regarding the calculation of $\mathrm{Im}\,\mathcal{G}(t_i, t_j)$ at step $i_0 + 1$.

Integral	Domain	Non-zero elements	Additions	Multiplications
$\int_0^{t_i} \mathrm{d}t_k$	$j \leqslant i_0(N-1)$	N	$i_0 N$	$(2i_0+1)N$
$\int_0^{t_j} \mathrm{d}t_k$	$j \leqslant i_0(N-1)$	0	$(j_0+1)N$	$(2j_0+1)N$
Total		$N+1$	$(i_0+j_0+1)N+1$	$2(i_0+j_0+1)N$
$\int_0^{t_i} \mathrm{d}t_k$	$j > i_0(N-1)$	$(i_0+1)(N-1)+1$	i_0	$(i_0+1)N$
$\int_0^{t_j} \mathrm{d}t_k$	$j > i_0(N-1)$	0	$(i_0+1)N$	$(2i_0+1)N$
Total		$(i_0+1)(N-1)+2$	$(i_0+1)(N+1)$	$(3i_0+2)N$

Table 3. Global communication and computation data regarding the calculation of $\mathcal{G}(t_i, t_j)$ at step $i_0 + 1$.

Equation	Floating-point numbers to be sent	Floating-point operations
$\mathrm{Re}\,\mathcal{G}(t, t')$	$(3i_0+1.5)N^2 - (3i_0+0.5)N$	$(5.5i_0+2)(i_0+1)N^2 + Ni_0$
$\mathrm{Im}\,\mathcal{G}(t, t')$	$(3i_0+1.5)N^2 - (3i_0+0.5)N$	$[(5.5i_0+2)(i_0+1)+i_0]N^2 + Ni_0$

Tables 1 and 2 summarize the number of floating-point operations performed in order to compute the non-vanishing matrix elements corresponding to a given i and j ($j < i$).

We can now calculate the ratio of communication to computation time, by noting that the numbers in the tables above get multiplied by N, corresponding to the number of collocation points in each time step and summing over the number of steps, i.e. we evaluate

$$N[\text{if } j > i_0(N-1)] + N \sum_{j_0=1}^{i_0} [\text{if } j \leqslant i_0(N-1)].$$

In table 3 we summarize all relevant estimates regarding the computation cost for a fixed value of i. In order to estimate the *total* communication and computation cost, respectively, these numbers must be multiplied by an additional factor of N, corresponding to the number of possible values of i in a time step. This factor is not relevant for estimating the communication overhead, but it must be remembered when one infers the sparsity of the corresponding system of equations.

To conclude, we observe that the communication to computation ratio approaches

$$\frac{1}{2(i_0+1)} \frac{T_{\mathrm{comm}}}{T_{\mathrm{calc}}}$$

for large values of i_0. Therefore, for this problem the communication overhead is reduced substantially in the later stages of the calculation. In practice, this ratio is actually much better, as we compute the functions $G(t, t')$ and $Q(t, t')$ on the fly, and this adds considerably to the computational effort. Finally, the sparsity of the resulting systems of equations goes to $2/(i_0 N)$ for large values of i_0 and N, which supports our choice for an iterative equation solver.

6. Conclusions

We have presented a numerical method suitable for solving nonlinear integral and integro-differential equations on a massively multiprocessor machine. Our approach is essentially a standard perturbative approach, where one calculates corrections to an initial guess of the solution. The initial guess is designed to satisfy the boundary conditions, and corrections are expanded in a complete basis of N Chebyshev polynomials on the grid of $(N + 1)$ extrema of $T_N(x)$, the Chebyshev polynomial of first kind of degree N. The spectral character of the convergence of the Chebyshev-expansion approach is the key element in keeping low the number of grid points. From a computational point of view, each iteration involves two stages, namely initializing the relevant matrices and solving the linear system of equations. Both stages can be rendered parallel in a suitable manner, and the efficiency of the code increases when applied to complicated multi-step, multi-dimensional problems.

The algorithm discussed in this paper represents the backbone of current investigations of the equilibrium and nonequilibrium properties of various phenomenological Lagrangians. In particular, we are interested in studying the properties of the chiral phase transition at finite density for a (2+1)-dimensional four-fermion interaction as well as the dynamics of two-dimensional QCD, with the ultimate goal of indirectly obtaining insights regarding the time evolution of a quark–gluon plasma produced following a relativistic heavy-ion collision.

Acknowledgments

The work of BM was supported in part by the US Department of Energy, Nuclear Physics Division, under contract no W-31-109-ENG-38. The work of RS was supported in part by the Natural Sciences and Engineering Research Council of Canada under grant no OGP0170170. Parallel calculations are made possible by grants of time on the parallel computers of the Mathematics and Computer Science Division, Argonne National Laboratory. BM would like to acknowledge useful discussions with John Dawson and Fred Cooper.

References

[1] Kerman A K and Koonin S E 1976 *Ann. Phys.* **100** 332
Jackiw R and Kerman A K 1979 *Phys. Lett.* A **71** 158
Guth A H and Pi S-Y 1985 *Phys. Rev.* D **32** 1899
Cooper F, Pi S-Y and Stancioff P 1986 *Phys. Rev.* D **34** 3831
Pi S-Y and Samiullah M 1987 *Phys. Rev.* D **36** 3128

[2] Boyanovsky D and de Vega H J 1993 *Phys. Rev.* D **47** 2343
Boyanovsky D, de Vega H J, Holman R, Lee D-S and Singh A 1995 *Phys. Rev.* D **51** 4419
Boyanovsky D, de Vega H J, Holman R and Salgado J 1996 *Phys. Rev.* D **54** 7570
Boyanovsky D, Cormier D, de Vega H J, Holman R, Singh A and Srednicki M 1997 *Phys. Rev.* D **56** 1939
Boyanovsky D, D'Attanasio M, de Vega H J, Holman R and Lee D-S 1995 *Phys. Rev.* D **52** 6805
Vautherin D and Matsui T 1997 *Phys. Rev.* D **55** 4492
Boyanovsky D, de Vega H J, Holman R and Salgado J 1998 *Phys. Rev.* D **57** 7388

[3] Cooper F and Mottola E 1987 *Phys. Rev.* D **36** 3114
Cooper F, Kluger Y, Mottola E and Paz J P 1995 *Phys. Rev.* D **51** 2377
Kluger Y, Cooper F, Mottola E, Paz J P and Kovner A 1995 *Nucl. Phys.* A **590** 581c
Lampert M A, Dawson J F and Cooper F 1996 *Phys. Rev.* D **54** 2213
Cooper F, Kluger Y and Mottola E 1996 *Phys. Rev.* C **54** 3298
[4] Wetterich C 1997 *Phys. Rev. Lett.* **78** 3598
Bettencourt L and Wetterich C 1998 *Phys. Lett.* B **430** 140
Bonini G F and Wetterich C 1999 *Phys. Rev.* D **60** 105026
[5] Aarts G, Bonini G F and Wetterich C 2001 *Phys. Rev.* D **63** 025012
[6] Cooper F, Habib S, Kluger Y, Mottola E, Paz J and Anderson P 1994 *Phys. Rev.* D **50** 2848
Cooper F, Dawson J F, Habib S, Kluger Y, Meredith D and Shepard H 1995 *Physica* D **83** 74
[7] Berges J and Cox J 2001 *Phys. Lett.* B **517** 369
Berges J 2001 *Nucl. Phys.* A **699** 847
Aarts G and Berges J 2002 *Phys. Rev. Lett.* **88** 041603
[8] Schwinger J 1961 *J. Math. Phys.* **2** 407
Bakshi P M and Mahanthappa K T 1963 *J. Math. Phys.* **4** 1
Bakshi P M and Mahanthappa K T 1963 *J. Math. Phys.* **4** 12
Keldysh L V 1964 *Zh. Eksp. Teor. Fiz.* **47** 1515 (Engl. transl. 1965 *Sov. Phys.–JETP* **20** 1018)
Zhou G, Su Z, Hao B and Yu L 1985 *Phys. Rep.* **118** 1
[9] Mihaila B, Dawson J F and Cooper F 1997 *Phys. Rev.* D **56** 5400
Mihaila B, Athan T, Cooper F, Dawson J F and Habib S 2000 *Phys. Rev.* D **62** 125015
Mihaila B, Dawson J F and Cooper F 2001 *Phys. Rev.* D **63** 096003
Blagoev K, Dawson J F, Cooper F and Mihaila B 2001 *Phys. Rev.* D **64** 125003
[10] Press W H, Teukolsky S A, Vetterling W T and Flannery B P 1992 *Numerical Recipes in FORTRAN: The Art of Scientific Computing* (New York: Cambridge University Press)
[11] Shaw R E 2000 A parallel algorithm for nonlinear Volterra integro-differential equations *Proc. 2000 ACM Symp. on Applied Computing* vol 1 p 86
[12] Mihaila B and Mihaila I 2002 Numerical approximations using Chebyshev polynomial expansions: El-gendi's method revisited *J. Phys. A: Math. Gen.* **35** 731
[13] El-gendi S E 1969 *Comput. J.* **12** 282
[14] Crisci M R, van der Houwen P J, Russo E and Vecchio A 1993 *J. CAM* **45** 169–80
Vecchio A 1993 Highly stable parallel Volterra Runge–Kutta methods *Rapp. Tecnico* no 102 Istituto per Applicazioni Della Matematica, Consiglio Nazionale Delle Ricerche, via P Castellino, 111, 80131 Napoli, Italy
[15] Sommeijer B P, Couzy W and van der Houwen P J 1992 *Appl. Numer. Math.* **9** 267
[16] IBM Redbooks 1999, RS/6000 SP: Practical MPI Programming IBM Corporation, Austin, TX
[17] da Cunha R D and Hopkins T R 1995 *Appl. Numer. Math.* **19** 33
da Cunha R D and Hopkins T R 1993 *Transput. Commun.* **1** 111
[18] Cooper F, Dawson J F, Habib S, Kluger Y, Meredith D and Shepard H 1995 *Physica* D **83** 74