Acta Univ. Sapientiae, Informatica **7,** 2 (2015) 216–238

DOI: 10.1515/ausi-2015-0020

Chaotic behavior of the lattice Yang-Mills on CUDA

Richárd FORSTER
Eötvös University
Faculty of Informatics
email: forceuse@inf.elte.hu

Ágnes FÜLÖP
Eötvös University
Faculty of Informatics
email: fulop@caesar.elte.hu

Abstract.
The Yang-Mills fields plays important role in the strong interaction, which describes the quark gluon plasma. The non-Abelian gauge theory provides the theoretical background understanding of this topic.The real time evolution of the classical fields is derived by the Hamiltonian for SU(2) gauge field tensor. The microcanonical equations of motion is solved on 3 dimensional lattice and chaotic dynamics was searched by the monodromy matrix. The entropy-energy relation was presented by Kolmogorov-Sinai entropy. We used block Hessenberg reduction to compute the eigenvalues of the current matrix. While the purely CPU based algorithm can handle effectively only a small amount of values, the GPUs provide enough performance to give more computing power to solve the problem.

1 Introduction

In particle physics there are more fundamental questions which demand the GPU, both of theoretical and experimental point of view.

Computing Classification System 1998: I.1.4
Mathematics Subject Classification 2010: 81T25
Key words and phrases: lattice gauge field, chaos in Hamiltoinan systems, parallel computing, GPU, CUDA

In the CERN NA61 collaboration one of most important research field is the quark-gluon examination. The aspects of theoretical physics includes the next current topics in the lattice field theory using GPU: strongly interacting Higgs sector, QCD hadron spectrum (eigenvalue distribution of he overlap Dirac operator).

We present a parallel algorithm, which enables to study the chaotic behaviour as Lyapunov spectrum of SU(2) Yang-Mills fields and the entropy-energy relation utilizing the Kolmogorov-Sinai entropy. We uses this method for large number of element of matrices to apply the CUDA platform particularly the eigenvalue of the monodromy matrix, which is an $\mathsf{f}\mathsf{x} \times \mathsf{f}$ sparse matrix ($\mathsf{f} = 24\mathsf{N}$).

The first step the non-Abelian gauge fields equation of motions is written by the lattice Hamiltonian SU(2) [2]. This system was solved by lattice process developed on the GPU [5]. The algorithm satisfies the constraint of the total energy and the unitarity, orthogonality of the suitable link variable on the 3 dimensional space.

In the next step we use the block Hessenberg reduction [11] to compute the required eigenvalues to determine the chaotic behaviour [8]. As it is described in [10] we are working with a hybrid system, that utilizes both the CPU and GPU for the most optimal performance. Thanks to this system it is possible to reach 2-3 times higher performance compared to the simple CPU based implementation of the same block Hessenberg reduction.

In the section 2 we introduce the basic concept of the non-Abelian gauge field. We describe the lattice regularization of Yang-Mills fields and SU(2) Hamiltonian to achieve the equations of motion in the section 3. We consider the chaotic description of the dynamics in the section 4, which contains chaos in the Hamiltonian systems and the lattice monodromy matrix method. The Hessenberg method is introduced in the section 5 to determine the eigenvalues of the monodromy matrix. The parallel hyprid Hessenberg algorithm is investigated in the section 6. The eigenvalue spectrum is determined by this parallel method and the numerical result is summarized in this section.

2 Gauge fields

The non-Abelian gauge[13] field plays important role in the theoretical particle physics. This theory based on principles of gauge invariance, which is derived from the Abelian gauge field to consider the principle of invariance under local gauge transformation.

In the electrodynamics field of charge e undergoes the local gauge transformation but derivative of this field does not transform like as the field itself, therefore we must introduce a field $A_\mu(x)$ ($\mu = 0, 1, 2, 3$) with gauge transformation property $A_\mu(x) \to A_\mu(x) + \partial_\mu \Lambda(x)$, with arbitrary $\Lambda(x)$ then we use it to construct a gauge invariant derivative of the original field of e, which transform just like this field. The gauge-invariant Lagrange can be constructed by these quantities. A dynamics for the gauge field is introduced by means of the Yang-Mills action:

$$S_{YM} = \frac{1}{4}\int d^4x F^a_{\mu\nu} F^a_{\mu\nu}, \tag{1}$$

where the $F^a_{\mu\nu}$ form is a component of an antisymmetric gauge field tensor in Minkowski space:

$$F^a_{\mu\nu} = \partial_\mu A^a_\nu - \partial_\nu A^a_\mu + g f^{abc} A^b_\mu A^c_\nu, \tag{2}$$

where $\mu\nu = 0, 1, 2, 3$ are space-time coordinates, the symmetry generators are labeled by $a, b, c = 1, 2, 3$ and g is the bare gauge coupling constant and f^{abc} are the structure constants of the continuous Lie group. The generators of this group fulfill the following relationship $[T^b, T^c] = i f^{bcd} T^d$. The equation of motion can be expressed by covariant derivative in the adjoint representation:

$$\partial^\mu F^a_{\mu\nu} + g f^{abc} A^{b\mu} F^c_{\mu\nu} = 0. \tag{3}$$

The Yang-Mills action contains cubic and quartic self-interaction terms. The original article[14] was published by Yang and Mills in 1954.

3 Lattice Yang-Mills fields

We will describe the lattice regularization on the Euclidean continuum to the hyper-cubic lattice [9].

The shortest non-zero distance on a hyper-cubic lattice is the lattice spacing a.

These are group elements which are related to the Yang-Mills potential A^c_i:

$$U_{x,i} = \exp(a A^c_i(x) T^c), \quad \text{where} \quad T^c \quad \text{is a group generator.} \tag{4}$$

For SU(2) symmetry group these links are given by the Pauli matrices τ, where $T^c = -(ig/2)\tau^c$. The indices x, i denote the link of the lattice which starts at the 3 dimensional position x and pointing into the nearest neighbour in direction i, $x+i$. We consider the collection of all link variables as the lattice gauge field.

3.1 Wilson action

We consider the construction of a gauge invariant action for the gauge field.

The non-Abelian gauge field strength can be expressed by the oriented plaquette i.e. product of four links on an elementary box with corners $(x, x+i, x+i+j, x+j)$:

$$U_{x,ij} = U_{x,i}U_{x+i,j}U_{x+i+j,-i}U_{x+j,-j}, \tag{5}$$

where $U_{x,-i} = U^{\dagger}_{x-i,i}$ and we will use this notation $U_p \equiv U_{x,ij}$.

The Wilson action is defined for pure lattice gauge theory [4]

$$S[U] = \sum_p S_p(U_p) \tag{6}$$

with the plaquette term:

$$S_p(U) = \beta(1 - \frac{1}{N}\mathrm{ReTr}U), \tag{7}$$

for SU(2) and β is a constant. Here the sum over all plaquettes p is meant to include every plaquette only with one orientation.

The Wilson action is gauge invariant since $\mathrm{Tr}U'_p = \mathrm{Tr}U_p$ and it is real and positive.

We consider the question, in which sense Wilson action for SU(2) is related to the Yang-Mills action for gauge fields on the continuum. Because $A_\mu(x)$ a Lie algebra values vector field. The expression (4) is extended by a, then the Wilson action is the following:

$$S = -\frac{\beta}{4N}\sum_x a^4 \mathrm{Tr}F_{\mu\nu}(x)F^{\mu\nu}(x) + O(a^5). \tag{8}$$

Thus the leading term for small 'a' coincidences with the Yang-Mills action if we set $\beta = \frac{2N}{g^2}$, where g identifies the bare coupling constant of the lattice theory.

The coupling on space-like and time-like plaquettes are no longer equal in the action:

$$S = \frac{2}{g^2}\sum_{p_t}(N - \mathrm{tr}(U_{p_t})) - \frac{2}{g^2}\sum_{p_s}(N - \mathrm{tr}(U_{ps})). \tag{9}$$

The time like plaquette is denoted by U_{p_t} and space like U_{p_s}.

Consider the path is a closed contour i. e. Wilson loop, it is invariant under gauge changes and independent of the starting point. The product of such group elements along the closed line is a gauge covariant quantity, the trace over such products are invariant. Because the U_{p_t} can be series expansion by a_t:

$$U_{p_t} = U(t)U^\dagger(t+a_t) = UU^\dagger + a_t U\dot{U}^\dagger + \frac{a_t^2}{2}U\ddot{U}^\dagger + \dots \tag{10}$$

$N - \mathrm{tr}(U_{p_t}) = -\frac{a_t^2}{2}\mathrm{tr}(U\ddot{U}^\dagger)$ up to $O(a_t^3)$ correction, where $UU^\dagger = 1$.

Therefore the homogenous non-Abelian gauge action:

$$\Delta S_H = \frac{2}{g^2}\left(\frac{a_t^2}{2}\sum_i \mathrm{tr}(\dot{U}_i\dot{U}_i^\dagger) - \sum_{ij}(N - \mathrm{tr}(U_{ij}))\right). \tag{11}$$

The Scaled Hamiltonian was derived in the next form:

$$a_t H = \frac{2}{g^2}\left(\frac{a_t^2}{2}\sum_i \mathrm{tr}(\dot{U}_i\dot{U}_i^\dagger) + \sum_{ij}(N - \mathrm{tr}(U_{ij}))\right). \tag{12}$$

The indices x, i denotes the link of the lattice starting at the 3 dimensional position x and pointing into the i-th direction.

3.2 SU(2) Hamiltonian

We study the real time classical evolution of the next Hamiltonian for SU(2) [3] [2]:

$$H = \sum_{x,i}\left(\frac{1}{2}\langle \dot{U}_{x,i}, \dot{U}_{x,i}\rangle + \left(1 - \frac{1}{4}\langle U_{x,i}, V_{x,i}\rangle\right)\right). \tag{13}$$

The complement variable $V_{x,l}(U)$ is constructed from triple product of links, which is complete to considere every link x, i to an elementary plaquette:

$$V_{x,l} = \frac{1}{4}\sum_{\left(\substack{(l,s):\{(i,j),(k,j),\\(-i,j),(-k,j)\}}\right)} U_{x+l,s}U^\dagger_{x+l+s,-l}U^\dagger_{x+l,-l}, \quad \text{where} \tag{14}$$

i, j, k note the unit vectors of three dimensional lattice. The canonical variable assigns by $P_{x,i} = \dot{U}_{x,i}$. We will denote the single link $U_{x,i}$ with U. Quaternion representation (for one link):

$$U = u_0 + i\tau^a u^a \qquad U = \begin{pmatrix} u_0 + iu_3, & iu_1 + u_2 \\ iu_1 - u_2, & u_0 - iu_3 \end{pmatrix}, \tag{15}$$

where, τ^a is the Pauli matrix. The lattice equation of motion is derived by canonical variable from this Hamiltonian.

3.3 Lattice equations of motion

The Hamiltonian equation of motion is solved with dt discrete time steps. This algorithm satisfies the Gauss law and the constraint of total energy[1]. We will denote single link $U_{x,i}$ in time t with U_t.

$$\begin{aligned} U_{t+1} - U_{t-1} &= 2\Delta t(P_t - \varepsilon U_t) \\ P_{t+1} - P_{t-1} &= 2\Delta t(V(U_t) - \mu U_t + \varepsilon P_t), \quad \text{where} \end{aligned} \tag{16}$$

$$\varepsilon = \frac{\langle U_t, P_t \rangle}{\langle U_t, U_t \rangle}, \quad \mu = \frac{\langle V(U_t), U_t \rangle + \langle P_t, P_t \rangle}{\langle U_t, U_t \rangle} \tag{17}$$

The ε, μ means the Lagrange multipliers and the symmetry SU(N) is fulfilled by the next expressions: $\langle U_t, U_t \rangle = 1$ (unitarity) and $\langle U_t, P_t \rangle = 0$ (orthogonality).

3.3.1 Implicit-Explicit-Endpoint algorithm

We apply these notions

$$P' = P_{t+1} \qquad P = P_t.$$

The Implicit-Explicit-Endpoint recursion algorithm:

$$P' = P + (V - \mu U + \varepsilon P') \tag{18}$$
$$U' = U + (P' - \varepsilon U), \tag{19}$$

where μ, ε are the Lagrange multipliers.

In the next section we study the nonlinearity of the Yang-Mills fields, which is described by the chaotic theory[12]. Instead of the classical rescaling solution we apply the monodromy matrix method, which can describe the gauge field evolution, in this case the short and long time behaviour. The question of ergodization is addressed via the Kolmogorov-Sinai entropy.

4 Measurement of chaos

We consider the description of dynamical systems. One of these is the Poencare map.

$$x_{i+1} = P(x_i) \quad i = 0, 1, 2 \ldots, \tag{20}$$

where we assign a hyper-surface $n-1$ dimension in the phase space n dimension and the trayectories are cutting it. The successive points of intersection can be given by this expression (20). Lyapunov exponent is defined by the next form:

$$\frac{d}{dt} x_i = F_i(x_1 \ldots x_n) \quad i = 1 \ldots N. \tag{21}$$

Let us introduce the quantity $\delta x_i(t)$:

$$\delta x_i(t) = x_i - \tilde{x}_i, \tag{22}$$

where $\delta x_i(t)$ means the distance between the two paths. The evolution of this notion is the following:

$$\frac{d}{dt}(\delta x_i) = \sum_{i=1}^{N} \delta x_k(t) \left(\frac{\partial F_i}{\partial x_i} \right)_{x_i = \tilde{x}_k(t)}. \tag{23}$$

The value of the distance $\delta x_i(t)$ is calculated by this expression:

$$D(t) = \left(\sum_{i=1}^{N} \delta(x_i(t))^2 \right)^{\frac{1}{2}}. \tag{24}$$

This quantity indicates the changing of the track distance $\delta \tilde{x}_i(t)$, where the paths were near at the beginning ($t = 0$). The maximal Lyapunov exponent follows:

$$L = \lim_{t \to \infty} \lim_{d(0) \to 0} \frac{1}{t} \ln \frac{D(t)}{D(0)}. \tag{25}$$

If $L > 0$, then the motion becomes chaoticity. We mention the rescaling method [7] briefly. Let us suppose there is a point q(0) in the phase space and the vectors $v_i, i = 1 \ldots v_K$ in the tangent space $T_{q(0)}$, we solve the equations of motion in phase space. We obtain q(t) and $v_i \in T_{q(t)}$ under the parallel evolution of the paths for a small perturbation of the initial condition in the tangent space. The Gram-Schmidt ortogonalisation is applied to determinate the tangent vector ν_i on the interval $k\tau$. The scaling vectors s_i are obtained by this procedure to calculate the Lyapunov exponents:

$$L_i = \lim_{n\to\infty} \sum_{k=1}^{n} \frac{\ln s_i^k}{\tau}, \tag{26}$$

where n is the number of iterations. One condition is to determine the tangent space at different points of phase space in this procedure. It is easy in the Euclidean space, but more difficult on the lattice gauge theory, because we need to perform the rescaling frequently.

The monodromy matrix method follows only one gauge field path for a long time evolution. This matrix is a linear stability matrix along a trayectory, which is solved by classical equation of motion and it provides the Lyapunov spectrum in the time evolution of field configuration on lattice.

4.1 Chaos in Hamiltonian system

An important part of the dynamical process is to provide the Hamiltonian function. This depends on the coordinate of space and momentum $H(q_i, p_i), i = 1 \ldots n$. The canonical conjugates variables are following

$$\dot{p_i} = -\frac{\partial H}{\partial q_i} \quad \dot{q_i} = \frac{\partial H}{\partial p_i}. \tag{27}$$

Let us given $q_i + \nu_i$, $p_i + \zeta_i$ nearby trajectories up to the linear approximation. Then the modified Hamilton function:

$$H^{'} = H + \left(\zeta_j \frac{\partial H}{\partial p_j} + \nu_j \frac{\partial H}{\partial q_j}\right). \tag{28}$$

The canonical variables are introduced:

$$\dot{\zeta}_i = -\frac{\partial}{\partial q_i}\left(\zeta_j \frac{\partial H}{\partial p_j} + \nu_j \frac{\partial H}{\partial q_j}\right), \tag{29}$$

$$\dot{\nu}_i = \frac{\partial}{\partial p_i}\left(\zeta_j \frac{H}{p_j} + \nu_j \frac{\partial H}{\partial q_j}\right). \tag{30}$$

The equations of motion can be written in the next form:

$$\begin{pmatrix} \dot{\zeta} \\ \dot{\nu} \end{pmatrix} = \begin{pmatrix} -\partial^2_{pq} H & -\partial^2_q H \\ \partial^2_p H & \partial^2_{pq} H \end{pmatrix} \begin{pmatrix} \zeta \\ \nu \end{pmatrix}. \tag{31}$$

If there exist at least one positive eigenvalue of this stability matrix, then the (ξ, ν) distance grows exponentially and the system is chaoticity.

The Lyapunov spectrum L_i is expressed in terms of the monodromy matrix's eigenvalues Λ_i [6]:

$$L_i = \lim_{T\to\infty} \frac{\int_0^T \Lambda_i(t)dt}{T}, \quad i = 1, \dots, f, \tag{32}$$

where $\Lambda_i(t)$ are the solutions of the characteristic equation:

$$\det[\Lambda_i(t)\mathbf{1} - M(t)] = 0. \tag{33}$$

at a given time t. Here M is the linear stability matrix, and f is the number of degrees of freedom. The discrete definition of the Lyapunov spectrum:

$$L_i' = \langle \Lambda_i \rangle = \frac{1}{n} \sum_{j=1}^{n} \ln \Lambda_i(t_{j-1}), \quad i = 1, \dots, f, \tag{34}$$

where t_j's are subsequent times along an evolutionary path of the gauge field configurations. In the conservative dynamics the Liouville's theorem is fulfilled:

$$\sum_{i=0}^{f} L_i = 0. \tag{35}$$

In the Hamiltonian system due to the conservation of the energy $L_i = -L_{f-i+1}$ is satisfied for every i. The Kolmogorov-Sinai entropy is expressed by Pesins formula:

$$h^{KS} = \sum_i L_i \Theta(L_i), \tag{36}$$

where $\Theta(x)$ being 1 for positive arguments and 0 otherwise. The dimension of h^{KS} is a rate (1/time) estimating the entropy:

$$S = \frac{h^{KS}}{\mathrm{Re}(L_0)N^3}. \tag{37}$$

4.2 Lattice monodromy matrix

We explain the elements of the matrix in this section. The monodromy matrix is the following:

$$M(t) = \begin{pmatrix} \frac{\partial \dot{U}}{\partial U} & \frac{\partial \dot{U}}{\partial P} \\ \frac{\partial \dot{P}}{\partial U} & \frac{\partial \dot{P}}{\partial P} \end{pmatrix}. \tag{38}$$

The matrix's elements of the lattice Hamiltonian for SU(2) are expounded:

$$\frac{\partial \dot{U}^a}{\partial U^b} = 0, \tag{39}$$

$$\frac{\partial \dot{U}^a}{\partial P^b} = \delta^{ab}, \tag{40}$$

$$\frac{\partial \dot{P}^a}{\partial U^b} = \frac{\partial V^a}{\partial U^b} - (\textstyle\sum_{c=1}^{N} U_c \frac{\partial V^c}{\partial U^b})U^a - V^b U^a - \sum_{c=1}^{N}(U_c V^c + P_c P^c)\delta^{ab}, \tag{41}$$

$$\frac{\partial \dot{P}^a}{\partial P^b} = -2P^b U^a, \qquad \text{where} \tag{42}$$

$$\frac{\partial V_k^{\alpha_q}}{\partial U^{\beta_q}} = \textstyle\sum_{l=1}^{\mathcal{N}} \frac{\partial V_k^{\alpha_q}(U_1,\ldots,U_{\mathcal{N}})}{\partial U_l^{\beta_q}}, \quad \text{ahol } \mathcal{N} = 12, \quad \alpha_q, \beta_q = 0,1,2,3. \tag{43}$$

The over-dots assign the derivative with respect to the scaled time t/a. They are providing information about the stability of trajectories in the neighbourhood of any point of an orbit in the (U, P) phase space. A small perturbation $(\delta U, \delta P)$, evolves in time governed by the monodromy matrix M. It is written by this form:

$$M(t) = \begin{pmatrix} 0 & 1 \\ \frac{\partial \dot{P}}{\partial U} & \frac{\partial \dot{P}}{\partial P} \end{pmatrix}. \tag{44}$$

The eigenvalues of this matrix can be classified as follows: for real and positive eigenvalues, neighbouring trajectories part exponentially and the motion is unstable. In the limit of large time we obtain the Lyapunov components from these eigenvalues to use the expression (34).

5 The eigenvalues of the monodromy matrix

The stability matrix is very rare and the number of element of matrix is very large. We applied the Hessenberg method [11] for the determination of the eigenvalues of the monodromy matrix, because the convergence of this method is very fast.

5.1 Balancing

The balancing procedure enables to take into account the sensitivity of the eigenvalues to rounding errors. The errors is proportional to the Euclidean norm of the matrix. The idea of balancing applies the similarity transformations to make corresponding rows and columns of the matrix have comparable norms, while leaving the eigenvalues unchanged. Balancing is a procedure of order N^2 operations.

We used the algorithm of Osborn. This process contains a sequence of similarity transformations by diagonal matrices D. The rounding errors was investigated during the balancing method, the elements of D are restricted to the powers of the radix, which base applied for floating-point arithmetic (i.e. 2 for most machines). The output is a matrix that is balanced in the norm, which is given by summing the absolute magnitudes of the matrix elements.

5.2 Reduction to Hessenberg form

First we reduced the matrix to a simplified form, it is called Hessenberg form, and the we applied an iterative procedure on the simpler matrix. Such structure can be accomplished by a sequence of Householder transformations, or other method, which is similar to Gaussian elimination with pivoting. Accordingly, the actual elimination procedure used is little bit different from Gauss elimination process.

Before the r-th stage, the original matrix $A \equiv A_1$ has become A_r, which was upper Hessenberg form in its first $r-1$ rows and columns. The r-th stage then contained the following operations:

- Search for the element of maximum magnitude in the r-th column below the diagonal. If it is zero, drop the next two "bullets" and the stage is done. Otherwise, suppose the maximum element was in row r'.

- The rows r' and $r+1$ are swapped (the pivoting procedure). To perform the permutation a similarity transformation, also swapped columns r' and $r+1$.

- For $i = r+2, r+3, \ldots N$ compute the multiplier
$$n_{i,r+1} \equiv \frac{a_{ir}}{a_{r+1,r}}.$$
Subtract $n_{i,r+1}$ times row $r+1$ from row i. To perform the elimination of the similarity transformation, we also add $n_{i,r+1}$ times column i to column $r+1$.

A total of $N - 2$ such stages are carried out.

When the magnitudes of the matrix elements changed by many orders, we rearranged the matrix so that the largest elements is situated in the top left-hand corner. This decreased the roundoff error (the reduction proceeds from left to right). The operation count is about $5N^3/6$ for large N.

5.3 The QR algorithm for real Hessenberg matrices

We took the following relations for the QR algorithm with shifts:

$$Q_s \cdot (A_s - k_s \mathbf{1}) = R_s \tag{45}$$

where Q is orthogonal and R is upper triangular matrix, and

$$A_{s+1} = R_s \cdot Q_s^T + k_s \mathbf{1} = Q_s \cdot A_s \cdot Q_s^T. \tag{46}$$

The QR transformation keeps the upper Hessenberg form of the original matrix $A \equiv A_1$ and the workload is $O(n^2)$ per iteration on such matrix. As $s \to \infty$, A_s converges to a term, where the eigenvalues are either isolated on the diagonal elements or they are eigenvalues of 2×2 submatrix on the diagonal. This shows a rapid convergence. The basic difference in this situation is that a nonsymmetric real matrix can have complex eigenvalues. This means that the eigenvalues may be complex for a good choices of the shifts k_s.

The complex arithmetic can be used in this process. We need here states that if B is a nonsingular matrix such that

$$B \cdot Q = Q \cdot H, \tag{47}$$

where Q is orthogonal and H is upper Hessenberg, then Q and H are fully determined by the first column of Q.

We used two step of the QR algorithm, either with two real shifts k_s and k_{s+1}, or with complex conjugate values k_s and $k_{s+1} = k^*$. This gives a real matrix A_{s+2}, where

$$A_{s+2} = Q_{s+1} \cdot Q_s \cdot A_s \cdot Q_s^T \cdot Q_{s+1}^T. \tag{48}$$

The Q's are calculated by the next expression:

$$A_s - k_s \mathbf{1} = Q_s^T \cdot R_s \tag{49}$$

$$A_{s+1} = Q_s \cdot A_s \cdot Q_s^T \tag{50}$$

$$A_{s+1} - k_{s+1}\mathbf{1} = Q_{s+1}^T \cdot R_{s+1}. \tag{51}$$

Let us used the equation (50), and the expression (51) can be written:

$$A_s - k_{s+1}\mathbf{1} = Q_s^T \cdot Q_{s+1}^T \cdot R_{s+1} \cdot Q_s. \tag{52}$$

Therefore, if we define

$$M = (A_s - k_{s+1}\mathbf{1}) \cdot (A_s - k_s\mathbf{1}) \tag{53}$$

equations (49) and (52) give

$$R = Q \cdot M, \tag{54}$$

where

$$Q = Q_{s+1} \cdot Q_s \tag{55}$$

$$R = R_{s+1} \cdot R_s. \tag{56}$$

The equation (48) can be rewritten:

$$A_s \cdot Q^T = Q^T \cdot A_{s+2}. \tag{57}$$

We search for an upper Hessenberg matrix H such that

$$A_s \cdot \overline{Q}^T = \overline{Q}^T \cdot H, \tag{58}$$

where $\overline{Q}$ is orthogonal. If $\overline{Q}^T$ has the same first column as Q^T, then $\overline{Q} = Q$ and $A_{s+2} = H$.

The first row of Q is determined as follows. The equation (54) presents that Q is orthogonal matrix and it triangularizes the real matrix M. Any real matrix can be triangularized with a sequence of Householder matrices P_1, P_2, ... P_{n-1}. Thus the matrix Q can expressed by $Q = P_{n-1} \ldots P_2 \cdot P_1$.

We need search for $\overline{Q}$, which is satisfying equation (58) whose first row is that of P_1. The Householder matrix P_1 is determined by the first column of M. Because A_s is upper Hessenberg, the equation (53) presents that the first column of M has the form $[p_1, q_1, r_1, 0, \ldots 0]^T$, where

$$\begin{aligned} p_1 &= a_{11}^2 - a_{11}(k_s + k_{s+1}) + k_s k_{s+1} + a_{12}a_{21} \\ q_1 &= a_{21}(a_{11} + a_{22} - k_s - k_{s+1}) \\ r_1 &= a_{21}a_{32}. \end{aligned} \tag{59}$$

Therefore

$$P_1 = \mathbf{1} - 2w_1 \cdot w_1^T,$$

where w_1 has only its first 3 elements nonzero. Proceeding in this way up to P_{n-1}, we can see that at each stage the Householder matrix P_r has a vector w_r, which is nonzero only in elements $r, r+1$ and $r+2$. These elements are calculated by the elements $r, r+1$, and $r+2$ in the $(r-1)$-st column of the current matrix.

The result is the next

$$P_{n-1} \cdots P_2 \cdot P_1 \cdot A_s \cdot P_1^T \cdot P_2^T \cdots P_{n-1}^T = H,$$

where H is upper Hessenberg matrix. Thus

$$\overline{Q} = Q = P_{n-1} \cdots P_2 \cdot P_1$$

and

$$A_{s+2} = H.$$

The shifts of the beginning at each stage are formed to the eigenvalues of the 2×2 matrix in the bottom right-hand corner of the current A_s.

This gives

$$\begin{aligned} k_s + k_{s+2} &= a_{n-1,n-1} + a_{nn} \\ k_s k_{s+1} &= a_{n-1,n-1} a_{nn} - a_{n-1,n} a_{n,n-1}. \end{aligned} \tag{60}$$

Substituting the expression (60) in the equation (59) we get

$$\begin{aligned} p_1 &= a_{21} \{[(a_{nn} - a_{11})(a_{n-1,n-1} - a_{11}) - a_{n-1,n} a_{n,n-1}]/a_{21} + a_{12}\} \\ q_1 &= a_{21}[a_{22} - a_{11} - (a_{nn} - a_{11}) - (a_{n-1,n-1} - a_{11})] \\ r_1 &= a_{21} a_{32}. \end{aligned} \tag{61}$$

We reduce possible roundoff, when there are small off-diagonal elements. Finally, we perform a double QR step we constructed the Householder matrices P_r $r = 1, \ldots n-1$.

For P_1 we applied p_1, q_1 and r_1, which were given by expressions (61). The remaining matrices, p_r, q_r and r_r were calculated by the $(r, r-1)$, $(r+1, r-1)$, and $(r+2, r-1)$ elements of the current matrix. The number of arithmetic operations can be decreased by writing the nonzero elements of the $2w \cdot w^T$ part of the Householder matrix in the form

$$2w \cdot w^T = \begin{bmatrix} (p \pm s)/(\pm s) \\ q/(\pm s) \\ r/(\pm s) \end{bmatrix} \cdot [1 \quad q/(\pm s) \quad r/(p \pm s)],$$

where

$$s^2 = p^2 + q^2 + r^2.$$

If we proceed in this way, convergence is usually very fast, which is need to control from step to step. There are two possible ways of terminating the iteration for an eigenvalue. First, if $a_{n,n-1}$ becomes 'negligible', then a_{nn} is an eigenvalue. We can then delete the n-th row and column of matrix and find the next eigenvalue. Otherwise $a_{n-1,n-2}$ may become negligible. Then the eigenvalues of the 2×2 matrix in the lower right-hand corner may be taken to be eigenvalues. We delete the n-th and $(n-1)$-th rows and column of the matrix and continue the process. The operation count for the QR algorithm described here is $\sim 5k^2$ per iteration, where k is the current size of the matrix.

In the next (6.) Section the significant question is the parallelisation of the Hessenberg method in rare matrix.

6 Parallel hybrid Hessenberg method

In [5] we used the CUDA platform to develop a parallel version of the Yang-Mills algorithm for lattice calculations. Here we give the details how we have moved forward from there by applying parallelism to calculate the eigenvalues of the monodromy matrix (4.1. subsection), which helps to show the chaos in the non-Abelian gauge field theory.

6.1 Main idea

Examining the Hessenberg method we can easily differentiate parts that has more computational intensive tasks, while others are not so performance sensitive. Hence the Block Hessenberg Algorithm is used. In this instead of taking the whole matrix as the input of the transformation process we divide it into smaller blocks. We take these blocks and calculate the Householder vector for each column in that block and with it update the consecutive columns. When finished we use the accumulated Householder transformations to update the rest of the whole matrix. We repeat this until we update all the blocks. This way with the accumulated Householder transformations in overall less matrix multiplications will be used compared to the original Hessenberg Algorithm in which case we always have to update every column with the calculated Householder vector.

6.1.1 Block householder algorithm

To calculate the Householder vectors we use the following [10]:

$$\nu = (\frac{1}{A_{k+1,k+\sigma}}) \tag{62}$$

$$\sigma = sign(A_{k+1,k}\|x\|_2) \tag{63}$$

$$V_k = [\nu_1, \nu_2, ..., \nu_k] \tag{64}$$

Compact-WY representation of the k Householder transformations:

$$\begin{aligned}(I - \nu_1\nu_1^T)...(I - \nu_k\nu_k^T) &= I - V_k T_k V_k^T \\ A := A(I - V_L T_L V_L^T) &= A - Y_L V_L^T \\ A &:= (I - V_L T_L V_L^T)A\end{aligned}$$

$$T_k = \begin{bmatrix} T_{k-1} & -\tau_k T_{k-1} V_{k-1}^T \nu_k \\ 0^T & \tau_k \end{bmatrix} \tag{65}$$

$$Y_k = AV_kT_k = \begin{bmatrix} Y_{k-1} & \tau_k(-Y_{k-1}V_{k-1}^T\nu_k + A\nu_k) \end{bmatrix} \tag{66}$$

We initialize V, T and Y as follows:

$$\begin{aligned} V_1 &= [\nu_1] \\ T_1 &= [\tau_1] \\ Y_1 &= [AV_1T_1] \end{aligned}$$

Update formula for one column of a block

$$A_{*,k} := A_{*,k} - Y_k((V_k)_{k,*})^T \tag{67}$$

$$A_{*,k} := (I - V_k T_k^T V_k^T) A_{*,k} \tag{68}$$

Update formula for the rest of the matrix

$$A_{*,L+1:n} := A_{*,L+1:n} - Y_L((V_L)_{L+1:n,*})^T \tag{69}$$

$$A_{*,L+1:n} := (I - V_L T_L^T V_L^T) A_{*,L+1:n} \tag{70}$$

With the hybrid implementation we extend the algorithm to the GPU as much as reasonably possible. The GPU will need a high amount of data to be able to achieve high utilization [15] and thus high performance, so the low on data parts of the calculation are kept on the CPU while the intensive matrix multiplications are pushed to the GPU.

Before commencing any calculations we upload the matrix into the GPUs memory, after that we follow the next steps for the k^{th} block:
For every column (i) in the block:

1. Compute the Householder vector (v), the i^{th} column of V (CPU) [eq. 62,63,64]

2. Update T and Y matrices (CPU) [eq. 65,66]

3. Update the next column (CPU) [eq. 67,68]

   After updating the block:

4. Update the rest of the matrix with V,Y,T (GPU) [eq. 69,70]

5. Copy over the next block to the CPU as it has been updated on the GPU

6. Continue the reduction

6.2 Restrictions

The monodromy matrix can become very big as we increase the N parameter of the lattice, thus requiring a lot of memory, which can go up to the TB range. Because of this right now reasonable results can be achieved only up to $N = 6$.

6.3 Implementation

For implementation and testing we have used a GeForce GTX 980M with compute capability 5.2 (Table 1) and an Intel Core i7-4710HQ CPU that does not have an IGP (Table 2).

As the starting point the original matrix for which we would like to compute the Hessenberg form will be uploaded to the GPU. After this as we compute the new Hessenberg vectors in the V matrix trough the blocks and update the T, Y matrices we are providing every element for the GPU to update our

	GeForce GTX 980M
Technical Specifications	Compute Capability 5.2
Transistors (Million)	5200
Memory (GB)	4
Memory Bandwidth (GB/s)	160
GFLOPs	3189
TDP (watts)	125

Table 1: GeForce GTX 980M technical specifications.

	i7-4710HQ
Transistors (Million)	1400
Connected memory (GB)	24
Memory Bandwidth (GB/s)	25.6
GFLOPs	422
TDP (watts)	47

Table 2: Core i7-4170HQ technical specifications.

original matrix. We copy the V, T, Y matrices to the GPU and using matrix-matrix multiplication we do the update. After this as the matrix has been changed the next block will be copied over to the CPU side. We choose the blocks to be 32 columns wide.

For doing the matrix multiplications in parallel on the GPU we use the NVIDIA developed CUBLAS library's *cublasdgemm* function, while on the CPU we use LAPACK with Intel MKL BLAS.

6.3.1 Comparison of the theoretical performance

Here we provide a comparison between the GPU's and CPU's achievable performance based on the GFLOPS and TDP values.

If we look at the computational power of the used processors we can see that the GPU is 7.5 times higher than what the CPU can provide. The GFLOPS of the CPU was calculated using the following formula:

$$\mathsf{GFLOPS} = \mathsf{cores} * \mathsf{clock} * \frac{\mathsf{FLOPs}}{\mathsf{cycle}} / 1000$$

The clock rate of a Core i7-4710HQ on full load with all 4 cores activated is

3.3 GHz and the maximum FLOPs/cycle is 32 [16], thus the maximum performance number in Table 2. If we would like to reach the GPU's performance with the actual CPU architecture, this means we will need 7.5 times more power. This means if we stay on the Haswell architecture and we will just try to increase the throughput we will reach a TDP of 352.5 watts. This leads us to the conclusion if we would like to have the same performance on the CPU that we have on the GPU will need 2.82 times more power for the CPU. This will also mean that the physical limitations will not hold back the increase of clock rate and power which can never be true, leaving the GPUs the most efficient processors.

6.4 Numerical results

To calculate the eigenvalues of the monodromy matrix we used the Hessenberg method. To make the process more efficient we applied the block Hessenberg model to be able to utilize parallelization.

For overall testing the following system was used (Table 3):

CPU	GPU	OS	Compiler	CUDA version
Intel Core i7 4710HQ	GeForce GTX 980M	Windows 10 Pro	Visual C++ 2013	7.0

Table 3: The used system's specification.

The numerical results fulfill the physical principle, the constraint value of the physical quantity remains constant during the time evolution of the equation of motion. The Lyapunov Spectrum (Figure 2) justifies the existence of the chaotic motion in the Yang-Mills fields.

We compared the runtime of the CPU to the GPU (Figure 1), the GPU gives substantially better results as we increase the available work. Evaluating the same lattice size the runtime on the GPU shows acceleration of a magnitude of 3.

The Kolmogorov-Sinai entropy (Figure 3) is obtained from the evolution eigenvalues of the monodromy matrix as functions of the scaled energy. These results gives good approximation for an ideal gas (S logE).

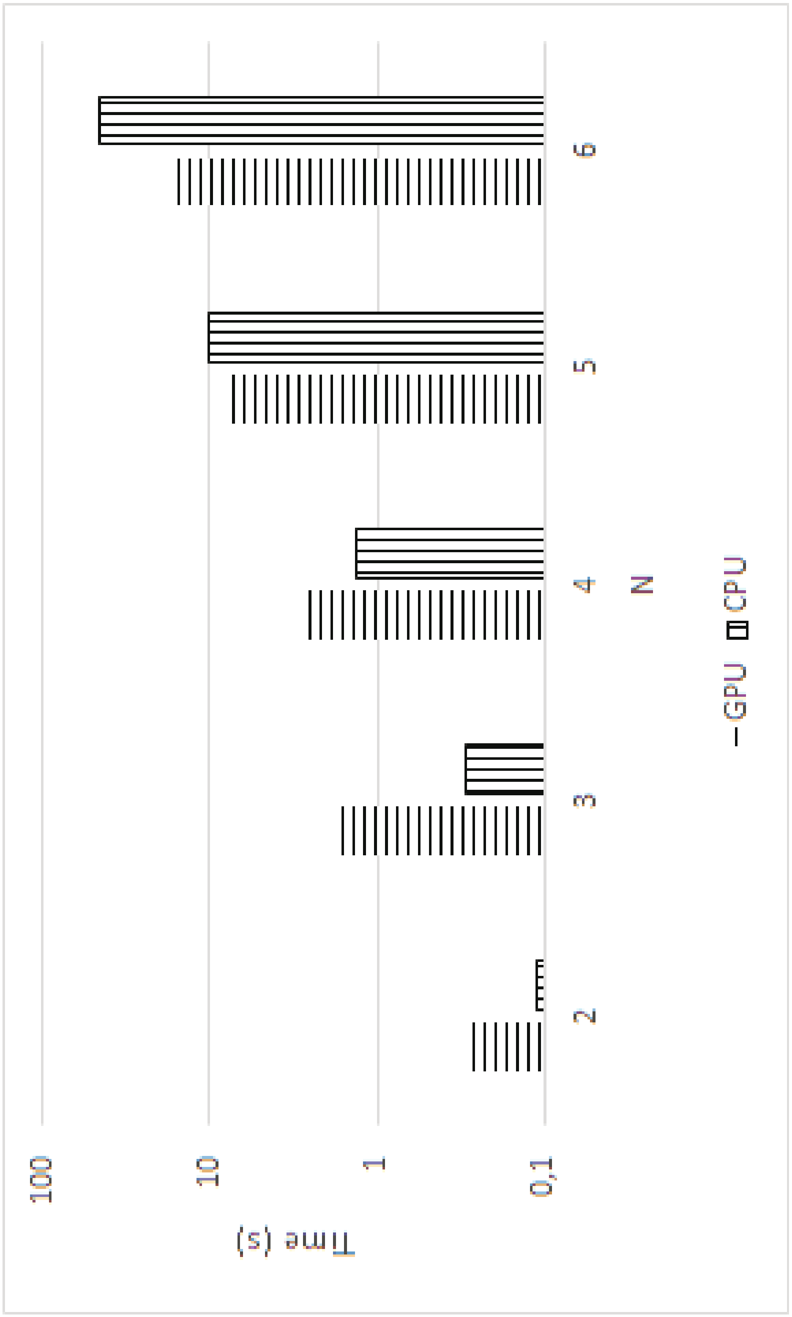

Figure 1: Runtime on the CPU and on the GPU with $\mathsf{N} = 2, 3, 4, 5, 6$.

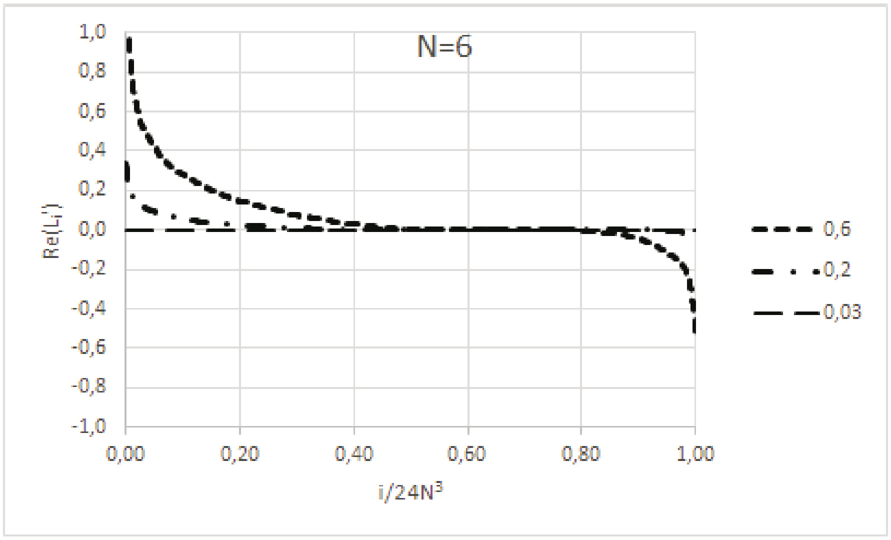

Figure 2: The Lyapunov spectrum on N=6.

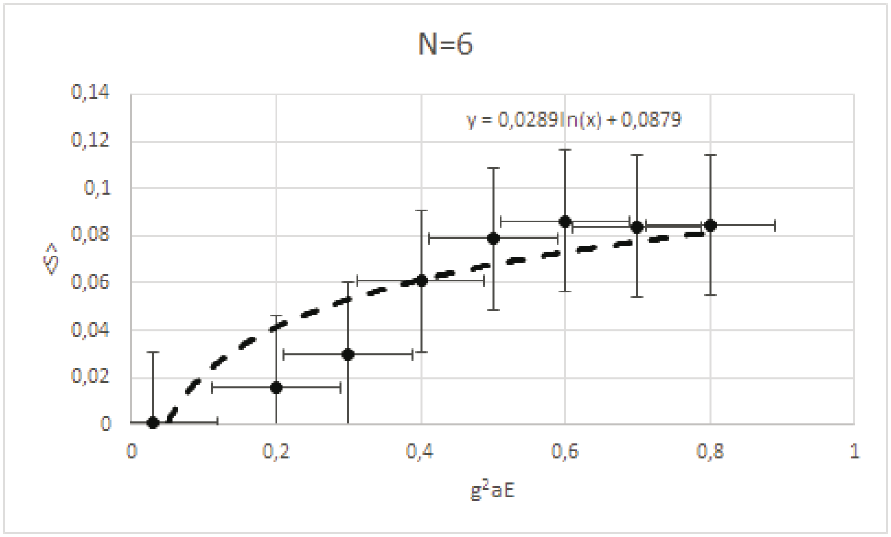

Figure 3: The Kolmogorov-Sinai entropy on N=6.

7 Summary

As the GPUs are becoming more powerful with each new architecture it becomes easier to modify the existing applications and algorithms to be parallel. In our case the Block Hamilton algorithm was able to achieve a 3 fold speed up compared to the CPU version, while computing the eigenvalues of the monodromy matrix.

By moving from CPU to GPU the eigenvalues are the same, thus keeping the physical principles valid. Physical constant quantities remains constraint while solving the equation of motion by parallel algorithm, such as the total energy.

The performance of the GPUs make it possible to calculate eigenvalues of the monodromy matrix to evince the chaotic behaviour of Yang-Mills system.

References

[1] T. S. Biró, Conserving algorithms for real-time non-Abelian lattice gauge theories, *Int. Journ. of Modern Phys. C* **6** (1995) 327–344. ⇒221

[2] T. S. Biró, A. Fülöp, C. Gong, S. Matinyan, B. Müller, A. Trajanov, Chaotic dynamics in classical lattice field theories,*165th WE-Heraeus Seminar on Theory of Spin Lattices and Lattice Gauge Models*, 14–19 Oct 1996. Bad Honnef, Germany, *Lec. Notes in Physics* **494** (1997) 164–176. ⇒217, 220

[3] T. S. Biró, C. Gong, B. Müller, A. Trayanov, Hamiltonian dynamics of Yang-Mills fields on a lattice, *Int. Journ. of Modern Phys. C* **5** (1994) 113–149. ⇒ 220

[4] M. Creutz, *Quarks, Gluons and Lattices*, Cambridge University Press, Cambridge CB2 1RP, 1983. ⇒219

[5] R. Forster, A. Fülöp, Yang-Mills lattice on CUDA, *Acta Univ. Sapientiae, Informatica*, **5**, 2 (2013) 184–211. ⇒217, 230

[6] A. Fülöp, T. S. Biró, Towards the equation of state of a classical SU(2) lattice gauge theory, *Phys. Rev. C* **64** (2001) 064902(5). *arxiv.org*. ⇒224

[7] C. Gong, Lyapunov spectra in SU(2) lattice gauge theory, *Phys. Rev D* **49** (1994) 2642–2645. ⇒222

[8] B. Müller, A. Trayanov, Deterministic chaos on non-abelian lattice gauge theory, *Phys. Rev. Letters* **68,** 23 (1992) 3387–3390. ⇒217

[9] I. Montvay, G. Münster, *Quantum fields on a lattice*, Cambridge University Press, Cambridge CB2 1RP, 1994. ⇒218

[10] J. Muramatsu, T. Fukaya, S. Zhang, Acceleration of Hessenberg Reduction for Nonsymmetric Eigenvalue Problems in a Hybrid CPU-GPU Computing Environment, *Intern. J. of Networking and Computing* **1,** 2 (2011) 132–143. ⇒217, 231

[11] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery, *Numerical Recipies in C*, Cambridge University Press, 2002. ⇒217, 225

[12] L. E. Reichl, *The Transition to Chaos*, Springer-Verlag, 1992. ⇒221

[13] S. Weinberg, *The Quantum Theory of Fields*, Cambridge University Press CB2 1RP, 1996 ⇒217

[14] C. N. Yang, R. Mills, Conservation of isotropic spin and isotopic gauge invariance, *Phys. Rev.* **96** (1954) 191–195. ⇒218

[15] CUDA C Programming Guide NVIDIA Corp., 2013, `http://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html`. ⇒232

[16] Technology Insight: Intel Next Generation Microarchitecture Code Name Haswell, IDF2012. ⇒234

Received: September 30, 2015 • Revised: December 29, 2015