EMPLOYING THE *S*-STEP CONJUGATE GRADIENT METHOD TO SOLVE SPARSE LINEAR SYSTEMS IN LATTICE QUANTUM CHROMODYNAMICS

Sabastian Atwood (Carleton DeTar)

Department of Physics and Astronomy

A significant problem in lattice quantum chromodynamics (LQCD) is solving the sparse linear systems $Ax = b$ that arise from the Dirac equation. The systems are large enough that the components need to be divided among several computer processors and the results communicated among processors. Currently, processors can perform computations faster than they can communicate with each other, so that reducing global communications presents an opportunity to reduce the time to solution.

A popular iterative algorithm for solving large, sparse linear systems is the conjugate gradient (cg) method. In each iteration, the method constructs a search direction conjugate to all previous directions and steps in that direction toward the solution. Each iteration requires the computation of 1 matrix product and the global sum (i.e., the global communication) of 2 dot products. Thus, for N iterations, there are N matrix products and $2N$ global sums. A variation of the conjugate gradient method known as the s-step method constructs and steps toward s many search directions in a single iteration and thus requires only N/s global sums. However, it must compute $s + 1$ matrix products per iteration, so that the total number of matrix products increases to $N(1+1/s)$. We wrote an s-step routine and compared its time-to-convergence to that of two conjugate gradient routines written for the purpose of solving LQCD systems. The first is a standard conjugate gradient routine that requires $2N$ global sums, and the second is a modified conjugate gradient routine called "fewsums" that requires only N global sums.

For the s-step algorithm, the time spent in matrix products and global sums is expected to be linear with respect to $1/s$. Figures 1 and 2 show test results confirming the linear relationships, with reduced chi-squared values of 1.5 and 0.3, respectively. The large error bars in Figure 2 reflect the varying levels of network interference on the computer cluster on which the jobs were run. For our relatively small number of 192 processors, the routine spent 95% of its time in matrix products and 2% of its time in global sums, so that the cost of extra matrix products exceeded the savings from reduced global sums. However, our model allows us to derive the condition that the s-step routine will be advantageous if $T_G > T_M/(s\text{-}1)$, where T_G and T_M are the total global-sum and matrix-product times, respectively, from the "fewsums" routine. The condition could be used to identify machines on which the s-step routine would be favorable over conjugate gradient.

Above a step size of 7, the s-step routine became unstable and would not converge. Further research could be done to improve the stability of the algorithm and thereby increase the maximum step size. According to our model and results, higher step sizes will further reduce both global-sum and matrix-product times.

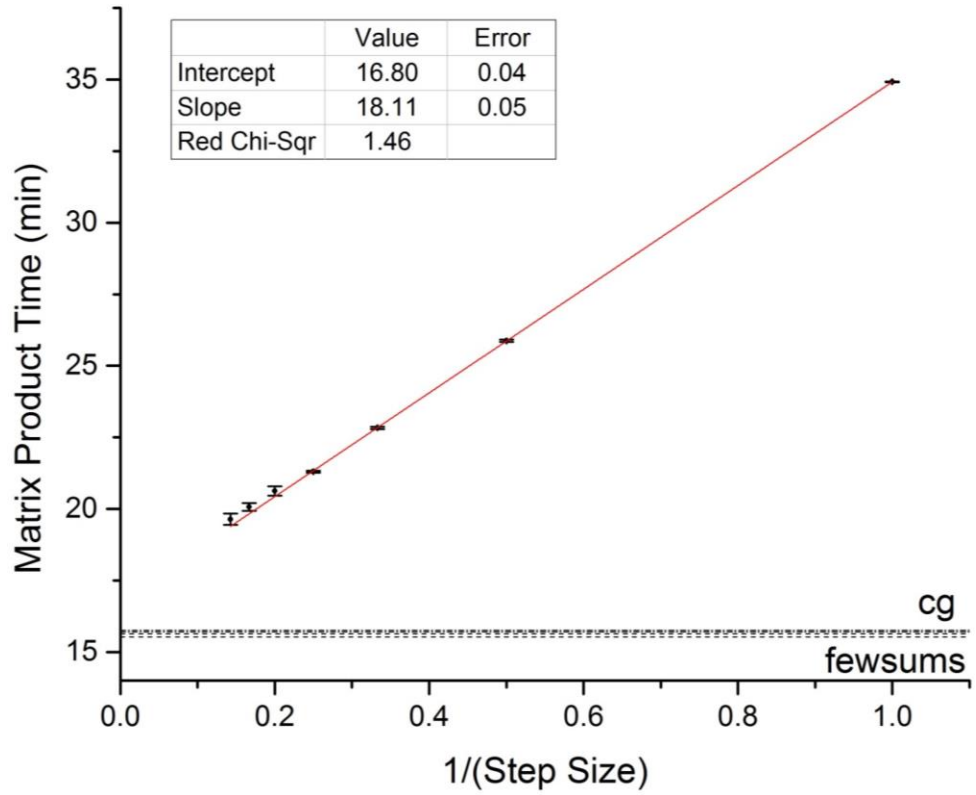

Figure 1: Time spent computing matrix products as a function of $1/s$.

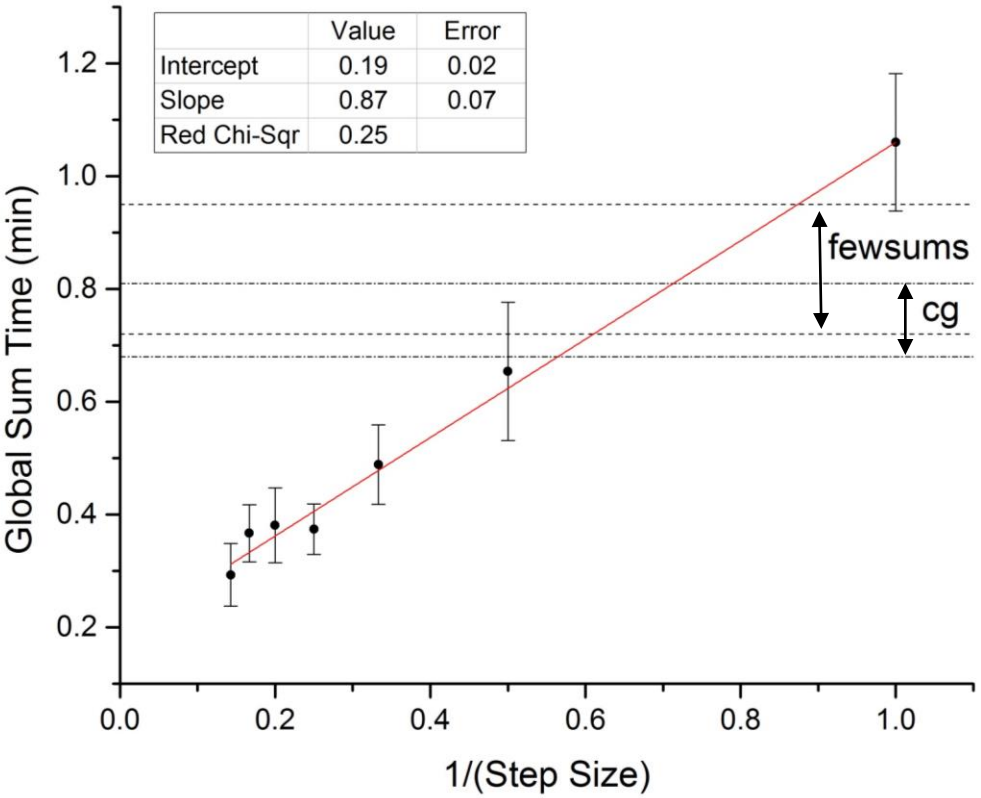

Figure 2: Time spent performing global sums as a function of $1/s$.