An R Package for Parallel Matrix Powers

Norm Matloff^{1*}, Jack Norman¹

1. University of California, Davis *Contact author: matloff@cs.ucdavis.edu

Keywords: matrix powers, parallel computation, Markov chains, graph connectedness, GPUs

Powers of square matrices are useful in a variety of contexts. Two examples in the statistics/data science field are calculating the stationary distribution of a discrete-time Markov chain and determining whether a graph is connected. Also, the exponential of a matrix M, defined as $e^M = \sum_{r=0}^{\infty} M^r/r!$, can be used to calculate the finite-time distribution of a continuous-time Markov chain (Stewart, 2009).

The CRAN package **expm** calculates matrix exponentials and powers. However, in many applications, the matrices involved are quite large, so parallel computation is needed. Our package **parmatpows** (http://heather.cs.ucdavis.edu/parmatpows) fills this need. Two parallel platforms are supported, multicore and GPU, based on the CRAN packages **Rdsm** and **gmatrix**, affording excellent speedups.

The package includes special functions for the applications mentioned above. The Markov chain example exploits the fact that for an irreducible, aperiodic chain with transition matrix P and stationary distribution π , $\lim_{n\to\infty} P(X_n=i)=\pi_i$, a fact that implies that $\lim_{n\to\infty} P^n$ is a matrix having each of its rows equal to π ; the column means of P^k will then provide a reasonable approximation to π for a suitably large k. In other words, finding the stationary distribution reduces to a matrix-powers problem. As with the matrix power function in **expm**, repeated squaring is used to compute a large enough power—P is squared to yield P^2 , which itself is then squared to yield P^4 and so on—thus reducing O(k) time to $O(\log_2 k)$.

It can be shown that an $n \times n$ graph with adjacency matrix A is connected if and only if $(A+I)^k$ consists of all positive values for all sufficiently large k (one need check only values of k through n-1). Thus connectedness can be determined again by repeated squaring.

Many mathematical operations used often in statistics, such as matrix inversion and QR factorization, are difficult to parallelize effectively on GPUs (Buckner, 2009). Thus it is not surprising that we found that in the Markov chain stationary distribution problem, use of matrix inverse on the GPU (gpusolve() in **gputools**) was not very effective.

But matrix multiplication is an exemplar data science application for GPU platforms, due to its regular pattern of data access and "embarrassingly parallel" nature, suggesting a matrix-powers approach to the stationary distribution problem. However, one needs to minimize data transfer back and forth between the CPU and GPU, which in turn means intermediate results must be saved between GPU kernel launches. In the context here, that means avoiding copying intermediate matrix powers if possible. This is not possible in **gputools**, but the **gmatrix** package, used here does allow saving on the GPU the result of a GPU matrix operation. (See also **RCUDA**, currently under development by D. Temple Lang and P. Baines.)

Similarly, on multicore platforms, our use of **Rdsm** is aimed at minimizing data copying, as it uses **big-memory** to directly access shared memory (though there may be some copying behind the scenes due to cache coherency actions).

References

Buckner, J., Justin Wilson, M. Seligman, B. Athey, S. Watson and F. Meng (2009). The gputools package enables GPU computing in R. *Bioinformatics*, 26, 1, 134-135.

Stewart, W. (2009). Probability, Markov chains, queues and simulation, Princeton University Press.