P. Koumoutsakos
ETH Zentrum, CLT E 13
CH-8092 Zürich

Fall semester 2018

Set 5 - Power Method, BLAS/LAPACK, OpenMP

Issued: October 26, 2018
Hand in (optional): November 02, 2018 23:59

Question 1: Power Method

The power method is an iterative technique for locating the dominant (largest) eigenvalue of a matrix. In addition, the power method also computes the associated eigenvector.

Consider the symmetric $N \times N$ matrix A, where the diagonal elements are given by $A[i,i] = \alpha i$ for $i = 1 \ldots N$ and the off-diagonal elements are random numbers drawn from a uniform distribution $\mathcal{U}[0,1]$ where $A[i,j] = A[j,i]$ for all $i \neq j$. Unless noted otherwise, use $\alpha \in \{1/8, 1/4, 1/2, 1, 3/2, 2, 4, 8, 16\}$ and $N = 1024$. Additionally, all results should be computed in double precision.

The power method produces a sequence of column vectors $\boldsymbol{q}^{(k)} \in \mathbb{R}^{N \times 1}$ given by

$$\boldsymbol{q}^{(k+1)} = \frac{A\boldsymbol{q}^{(k)}}{\|A\boldsymbol{q}^{(k)}\|_2} \tag{1}$$

If $\boldsymbol{q}^{(0)}$ is not deficient and the largest eigenvalue of A is unique, then $\boldsymbol{q}^{(k)}$ will converge to an eigenvector with eigenvalue $\lambda^{(k)}$.

a) Implement your own matrix multiplication program to calculate the dominant eigenvalue of the matrix A using the power method. Stop at the k-th iteration if the condition $|\lambda^{(k)} - \lambda^{(k-1)}| < 10^{-12}$ is satisfied. Use $\boldsymbol{q}^{(0)} = [1, 0, 0, \ldots]^T$ as the initial guess. Report the following:
   i) The dominant eigenvalue for all values of α.
   ii) The smallest and largest iteration numbers for a converged solution using the set of matrices computed with the corresponding α values. Report the α values that correspond to the smallest and largest iteration numbers as well.

b) Instead of a manual matrix-vector multiplication, use the CBLAS routines to perform the matrix operations of the power method. Write a program that allocates and initializes the matrix A, and computes the largest eigenvalue for the different values of α. Report the following:
   i) The dominant eigenvalue for all values of α. Report the smallest and largest iteration numbers for convergence and the corresponding α values as well.
   ii) Compute the time-to-solution (time to converged solution) of the CBLAS implementation, t_{power}, and of the manual matrix-vector multiplication implementation (previous subquestion), t_{manual}. Plot the speedup $S = t_{manual}/t_{power}$ as a function of α. If you observe a large speedup then examine your manual implementation and reason why.

iii) Report the time-to-solution for the CBLAS and the manual implementation for fixed $\alpha = 4$. Run the tests for the two matrix sizes $N = 4096$ and 8192.

c) By using the Power method, we can compute only the eigenvector corresponding to the largest eigenvalue of a diagonalizable matrix A. In this subquestion you will solve the full eigenvalue problem by computing the eigenvalues of matrix A using an appropriate routine provided by the LAPACK library. The Intel Math Kernel Library (MKL) includes a high-performance implementation of both BLAS and LAPACK libraries. In order to use the MKL on Euler, you have to load the module with `module load mkl`. After loading the module, you can include the header `#include <mkl_lapack.h>` to access the LAPACK routines.

   Write a program that allocates and initializes the same symmetric $N \times N$ matrix A as in the previous subquestions, and then calls the `LAPACKE_dsyev` routine of LAPACK to compute the full eigen solution of A. Report the following:

   i) The **two** dominant eigenvalues for each α.

   ii) The time required for your algorithm to converge, as a function of α.

   iii) Compute the time-to-solution of the full eigen solution, t_{ev_full}. Plot the speedup $S = t_{ev\_full}/t_{power}$ as a function of α.

d) Prove on paper that the initial guess converges to the largest eigenvector. Comment on the convergence behavior of the Power method with respect to α and relate your explanation to the result of your proof.

Question 2: OpenMP bug hunting

a) Identify and explain any *bugs* in the following OpenMP code. Propose a solution. Assume all headers are included correctly.

```
// assume there are no OpenMP directives inside these two functions
void do_work(const float a, const float sum);
double new_value(int i);

void time_loop()
{
    float t = 0;
    float sum = 0;

    #pragma omp parallel
    {

        for (int step=0; step<100; step++)
        {
            #pragma omp parallel for nowait
            for (int i=1; i<n; i++) {
                b[i-1] = (a[i]+a[i-1])/2.;
                c[i-1] += a[i];
            }

            #pragma omp for
            for (int i=0; i<m; i++)
                z[i] = sqrt(b[i]+c[i]);

            #pragma omp for reduction(+:sum)
            for (int i=0; i<m; i++)
                sum = sum + z[i];

            #pragma omp critical
            {
                do_work(t, sum);
            }

            #pragma omp single
            {
                t = new_value(step);
            }
        }
    }
}
```

b) Identify and explain any *improvements* that can be made in the following OpenMP code. Propose a solution. Assume all headers are included correctly.

```
void work(int i, int j);

void nesting(int n)
{
    int i, j;
    #pragma omp parallel
    {
        #pragma omp for
        for (i=0; i<n; i++)
        {
            #pragma omp parallel
            {
                #pragma omp for
                for (j=0; j<n; j++) {
                    work(i, j);
                }
            }
        }
    }
}
```