Parallel Homotopy Method for the Symmetric Eigenvalue Problem

Peter Reheis and Peter Yoon
Department of Computer Science, Trinity College, Hartford, CT

The Homotopy Method

Given a real symmetric matrix A of order n, consider the one parameter family of matrices:

$$A(t) = D + t(A - D) \quad t \in [0,1]$$

Where D is a block diagonal matrix matrix whose eigensystem is easy to find.

The eigenvalues of $A(t)$ are continuous functions of t. The graphs of these functions are referred to as eigencurves.

The Homotopy method proceeds as follows:

I. *Prediction*
For the initial prediction, 3rd order Taylor method is used to predict an eigenvalue. For subsequent predictions, Hermite interpolation is used.

II. *Corrections*
Rayleigh Quotient and Inverse Iteration are the methods used to correct out predicted eigenpair from (I).

III. *Checking*
When we are trying to follow the k^{th} eigencurve, there is always the chance that we may jump into the wrong eigencurve. To avoid this we compute $COUNT(\alpha)$, the number of eigenvalues less than α for A(t). This is done after prediction and checking to ensure that we are on the correct eigenpath.

IV. *Selection of Step Size*
In the first attempt, we always choose the step size $h = 1$. If any of the checking steps fail, the step size is cut in half and then we repeat the process over again until the eigenpair has been located.

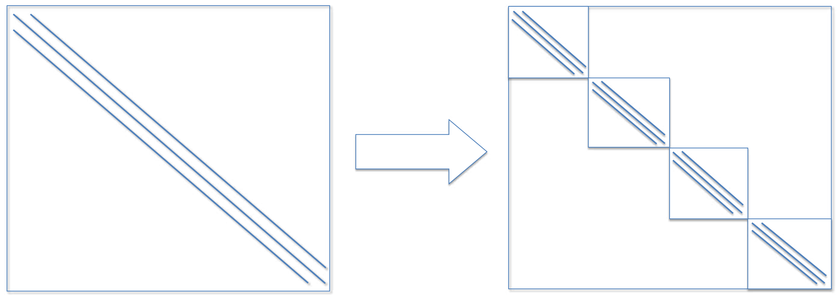
The Order Preserving Property

We can choose D such that $A(t) = D + t(A - D)$ is real symmetric tridiagonal with nonzero off-diagonal entries when $t \neq 0$. If we choose D to be diagonal, then it will have distinct eigenvalues. Thus there are n distinct curves that connect the eigenpairs of D to those of A. Since A(t) has distinct eigenvalues for $t \in [0,1]$, the eigencurves never cross each other. Then the k^{th} smallest eigenvalue of D is also the k^{th} smallest eigenvalue of A. Therefore, the Homotopy method can compute each eigenvalue independently of one another.

The Homotopy method can be applied to solve eigenvalue-eigenvector problems for symmetric tridiagonal matrices. It is often not of any interest to compute every eigenvalue of a matrix. Because the Homotopy method possess the order preserving property, the algorithm is able to compute any specific eigenvalue without the need for computing any other eigenvalues. Because of this property, the Homotopy method is also a highly parallel algorithm. By using CUDA and the cuBLAS and MAGMA libraries, we achieved up to 27x speed up in computation time. Numerical results show our method is highly efficient, especially for graded matrices.

A GPU-based Approach

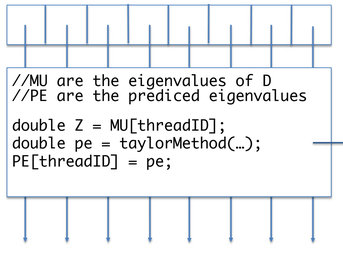
```
//Launch p blocks to obtain the submatrices of A to form D
//Solve the eigensystems of the blocks of D on the GPU
```



```
/*Tridiagonal Matrix A whose
eigenvalues we are trying to find.*/
```

```
/*magma_dsyevd_gpu(…) computes the
eigenvalues and eigenvectors of
each block.*/
```

```
/*Launch t threads to predict as
many eigenvalues*/
```



```
//MU are the eigenvalues of D
//PE are the prediced eigenvalues

double Z = MU[threadID];
double pe = taylorMethod(…);
PE[threadID] = pe;
```

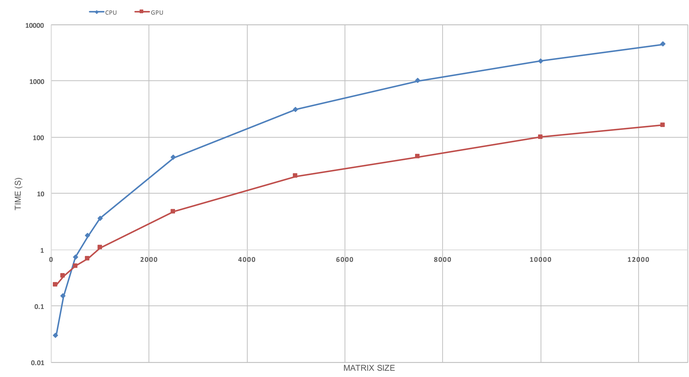
```
/*Taylor's Method to predict eigenvalues*/

/*Z1, Z2, Z3 are the eigenvalue derivatives
which were computed using dgemm() on the
GPU*/

/*Since all eigenvalues are predicted the
same way we can easily exploit this
parallelism*/

pe = Z+Z1*h+Z2*(h^2/2)+Z3(h^3/6)
```

```
/*Once all the predicted eigenvalues are collected, we can begin the correction process.*/

/*Rayleigh Quotient and Inverse Iteration take advantage of the highly optimized cuBLAS and
MAGMA libraries to significantly reduce computation time.*/

//Inverse Iteration

/*Solving the large sparse linear
system is computationally intensive and
requires significant amount of time for
large matrices*/

cublasDscal(…)
cublasDnrm2(…)//Scale and Normalize
magma_dgesv_gpu(…)//Solve linear system

    /*dgesv on the GPU significantly
    reduces computation time.*/
```

```
//Rayleigh Quotient Iteration

/*With a better approximate eigenvector,
RQI finds an eigenvalue*/

cublasDscal(…)
cublasDnrm(…)//Scale and Normalize


cublasDgemv(…)
cublasDdot(…)//Compute Rayleigh Quotient
    /*This eigenvalue undergoes
    correction until its residual is less
    than the prescribed accuracy.*/
```

Results

Our implementation of the Homotopy method was tested with the following: Intel® Xeon® E5-2620 CPU and NVIDIA® Tesla® K20c GPU with 64 GB main memory and 5GB GPU memory.



Matrix Size	100	250	500	750	1000	2500	5000	7500	10000	12500
Residual (Serial)	8.70E-15	2.18E-14	3.18E-14	3.19E-14	6.44E-14	1.61E-13	2.55E-13	3.94E-13	5.23E-13	4.71E-13
Residual (Parallel)	4.27E-15	1.51E-14	2.09E-14	2.03E-14	3.42E-14	9.45E-14	1.74E-13	1.81E-13	3.33E-13	3.85E-13

We compared our GPU-based implementation with a serial CPU-based implementation. The CPU-based implementation used LAPACK and BLAS routines to optimize performance for all matrix and vector computations Above are the total times to compute 20 middle eigenvalues as well as the maximum residuals. For smaller matrices, n<250, the GPU-based implementation was slower. However, when the order of the matrices grew larger than 10,000, there was over 20x speedup in computation time.

Future Work

- *A Multi-GPU and Multi-CPU implementation.*
- *Improve performance and accuracy on matrices with poorly separated and exterior eigenvalues.*
- *A method for dealing with bifurcation points where eigenvalues and eigenvectors are undefined on eigencurves.*

References

1. *T.Y. Li, N.H. Rhee, Homotopy Algorithm for Symmetric Eigenvalue Problems, 1989*
2. *L.J. Hwang, T.Y. Li, Parallel homotopy algorithm for symmetric large sparse eigenproblems, 1996*
3. *Andrzej Chrzeszczyk, Jakub Chrzeszczyk, Matrix computations on the GPU, CUBLAS and MAGMA by example, 2013*
4. *Timothy Sauer, Numerical Analysis, Prentice Hall, 2005*

Acknowledgements

This research was made possible by:
- *CUDA Teaching Center Program, NVIDIA*
- *Faculty Research Committee, Trinity College*