Applying OpenMP to eigenvalue computation

Emil Løvbak and Sahar Chehrazad

December 7th, 2020

Computing eigenvalues of a matrix A

- **Power method:** $x_k = Ax_{k-1}$
  x_k converges to an eigenvector which is multiplied with the corresponding eigenvalue in each iteration.
- **Subspace iteration:** $Q_k = AQ_{k-1}$
  We compute multiple eigenvalues by iterating with Q_k, containing columns of linearly independent vectors. We ensure independent columns of Q_k by performing a QR-decomposition with Gram-Schmidt after each iteration.

$Q_0 \leftarrow$ random_matrix()
for k $= 1 \ldots$ k_max **do**
  $\tilde{Q}_k \leftarrow AQ_{k-1}$
  $[Q_k, R_k] \leftarrow$ Gram-Schmidt$(\tilde{Q}_k)$
end for
eigenvalues $\leftarrow$ diagonal$(R_{\text{k_max}})$

Getting started

- Download the file `eigenvalues.cpp` from Toledo
- (On the VSC:) Load the module `Boost`
- Compile the program with `g++ -DNDEBUG -O3 -o eigenvalues eigenvalues.cpp`
- Run the program as `./eigenvalues <sizeOfA> <numberEigenvalues> <numberIterations>`
- Try to compute all eigenvalues of a 10x10 matrix with 5, 10 and 20 iterations.
- Now look at the code and try to understand how it works.

Parallelizing with OpenMP

- Identify the different loops in the program. Can they all be parallelized (efficiently)?
- What is the most computationally expensive part of the code? What does Amdahl's law tell us?
- Play around with making different loops parallel.
  - Consider two cases:
    - Computing all eigenvalues of a (reasonably) large matrix.
    - Computing 10 eigenvalues of a very large matrix `sizeOfA` ≥ 10000.
  - Verify that you get the same results as the serial code.
  - Check what gives the best improvement to the run-time.
- (extra) Play around with scheduling of the loops, do you notice a big difference?