CS 292F.300 Final project proposal

Ganesh Sankaran

TOTAL POINTS

1 / 1

QUESTION 1

1 Proposal submitted **1 / 1**

✓ **- 0 pts** **Correct**

This looks great. One suggestion: when you measure performance of the Kazmarcz solvers, don't only use running time, but also put in an ability to count the number of floating-point arithmetic computations (at least approximately). That way you can separate out the issues of the work complexity of the algorithm and the efficiency of the implementation (CPU vs GPU etc).

CS292F: Final Project Proposal

Collaborators: Albert Li and Ganesh Sankaran

We propose to experiment with randomized and non-randomized Kaczmarz solvers for linear systems Lx = b, where L is a graph Laplacian. As we discussed in lecture, such linear systems have applications in partial differential equations, circuits, and data analysis, among other areas in the sciences and engineering. In particular, we looked at networks of resistors, where we solve $Lv = i_{ext}$ to find the potentials on vertices under externally applied currents on edges.

Experiments: In general, we want to compare the performance of randomized and non-randomized Kaczmarz solvers for different sizes and structures of graphs against different linear solvers. In terms of "performance," we will examine the convergence behavior of the Kaczmarz solvers with and without randomization. For the randomized variant, we will also examine the convergence behavior under different row selection probability distributions, including random, uniform, non-uniform, directly proportional to the vertex degree, and inversely proportional to the vertex degree. The other linear solvers we will compare Kaczmarz to include pcg, minres, symmlq, and tfqmr, which MathWorks claims are designed (or work well) for symmetric matrices[1]. We may compare Kaczmarz to even more linear solvers depending on how the project progresses. If possible, we will try running Kaczmarz with GPU parallelization; however, from a cursory glance, it appears that this may not be useful because of the overhead of copying memory to and from the GPU as well as the lack of support for many operations, such as indexing of sparse GPU matrices.

Datasets: We will use three datasets. First, we will generate "toy" graphs based on classes we have covered in lecture, including complete, star, path, grid, etc. We will generate several graphs of different sizes for each class to observe how the scale and structure of the graph influences the performance of Kaczmarz solvers. Second, we will use graphs provided in .mat files either from the course GitHub repo or from Matlab (e.g. bucky). Third, we will use some larger undirected graphs from the SuiteSparse collection as a "stress test" of the Kaczmarz solvers.

Software Environment: We will use Matlab R2021a, with the latest versions of all associated packages, on a Windows machine. We are already implementing our own randomized and non-randomized Kaczmarz solvers, but for other linear solvers, we will use the pre-installed Matlab routines (such as minres() or symmlq()).

Hardware: We will use a lab machine with a 4.50 GHz six-core CPU, 16 GB of RAM, and a dedicated GPU with 8 GB of VRAM. We will try exploiting GPU parallelization provided by the Parallel Computing Toolbox in Matlab.

[1] https://www.mathworks.com/help/matlab/math/iterative-methods-for-linear-systems.html#f6-14578

1 Proposal submitted 1 / 1

✓ **- 0 pts Correct**

This looks great. One suggestion: when you measure performance of the Kazmarcz solvers, don't only use running time, but also put in an ability to count the number of floating-point arithmetic computations (at least approximately). That way you can separate out the issues of the work complexity of the algorithm and the efficiency of the implementation (CPU vs GPU etc).