MACM 316 - Computing Assignment 4

- Read the *Guidelines for Assignments* first.
- Submit a one-page PDF report to Canvas and upload your Matlab scripts (as m-files). Do not use any other file formats.
- Keep in mind that Canvas discussions are open forums.
- You must acknowledge any collaborations/assistance from colleagues, TAs, instructors etc.

Matrix exponentials

The Taylor series of the exponential function expanded around zero is

$$\exp(x) = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + ...$$

For an $n \times n$ matrix A we can define the matrix exponential as

$$\exp(A) = I + A + \frac{1}{2!}A^2 + \frac{1}{3!}A^3 + ...$$

where I is the $n \times n$ identity matrix. These exponentials have a number of uses but can be challenging to compute. (For an extensive analysis on the many ways to calculate the matrix exponential, see http://www.cs.cornell.edu/cv/researchpdf/19ways+.pdf)

One of the simplest ways to calculate the matrix exponential is to sum up the first k terms in the series defined above. This report will focus on analyzing the accuracy and efficiency of this algorithm. Download the file `CA4.mat` from Canvas. This contains the matrix A you will be testing your algorithm on. Load it with the following command:

```
load('CA4.mat');
```

The matrix A should now be in your workspace.

Make sure your algorithm works by running it for $k = 50$ and $k = 150$. Plot the results by using the commands:

```
imagesc(real(expAk));
colormap gray
```

It should be obvious whether or not your algorithm works.

Matlab has a built-in function for computing the matrix exponential; use this in your analysis of the error.

```
expA = expm(A);
err = norm(expA - expAk)/norm(expA);
```

(By default, `norm()` calculates the 2-norm of a vector or matrix.) Test your algorithm for $k = 10, 20, 30, ..., 150$ and measure both the error and computation time as functions of k. Provide plots of both.

How does computational time depend on k? Be as precise as possible. Count the number of flops in the algorithm; how does this compare with the computation time?

How does the error depend on k? Use a logarithmic y–axis and again be as precise as possible. How does round-off error affect the result?

Tips

- There are a few ways to implement the algorithm. Some will take significantly longer than others. Consider how the term added in the i–th step differs from the term added in the $(i-1)$–th step.
- The type of error I've suggested you use is the relative error in the 2-norm. You can speed up calculating it by pre-computing `norm(expA)`.
- You're comparing your flop count to the computational time of the algorithm. One easy way to do this is to plot them together. You may need to multiply the flop count by a small number to account for the computational time per flop.